НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«На правах рукопису»
УДК 004.77

«До захисту допущено»
Науковий керівник кафедри
_____ І.А. Дичка
«___»_____2019 р.

Магістерська дисертація

на здобуття ступеня магістра

зі спеціальності 121 Інженерія програмного забезпечення

на тему: «Метод маршрутизації транзакцій в мережі криптовалютних каналів з прихованою топологією»

Виконав:
студент ІІ курсу, групи КП-71мн
Жикін Юрій Сергійович _____

Керівник:
Доцент кафедри ПЗКС, к.т.н.,
Онай Микола Володимирович _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н.,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри АПЕПС, к.т.н.,
Смаковський Денис Сергійович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2019 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – другий (магістерський) за освітньо-науковою програмою

Спеціальність (спеціалізація) – 121 «Інженерія програмного забезпечення» («Програмне забезпечення комп'ютерних та інформаційно-пошукових систем»)

ЗАТВЕРДЖУЮ

Науковий керівник кафедри

_____ І.А. Дичка

«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Жикіну Юрію Сергійовичу

1. Тема дисертації «Метод маршрутизації транзакцій в мережі криптовалютних каналів з прихованою топологією», науковий керівник дисертації Онай Микола Володимирович, к.т.н., доцент, затверджені наказом по університету від «8» квітня 2019 р. №1075-С.

2. Термін подання студентом дисертації «17» травня 2019 р.

3. Об'єкт дослідження: процес маршрутизації транзакцій в мережі криптовалютних каналів в умовах відсутності інформації про її топологію.

4. Предмет дослідження: методи, способи, алгоритми та програмна реалізація маршрутизації транзакцій в мережах криптовалютних каналів.

5. Перелік завдань, які потрібно вирішити:
- провести вивчення предметної області дослідження, зокрема детально розглянути принципи роботи мереж криптовалютних каналів та загальні підходи до маршрутизації в комп'ютерних мережах;
- дослідити проблеми існуючого методу маршрутизації в мережі криптовалютних каналів Lightning;
- побудувати модель мережі, що виділяє класи вершин за функціями, які вони виконують, і на її основі сформулювати проблему відкритої фінансової топології;
- розробити метод маршрутизації транзакцій в мережі криптовалютних каналів, інформація про топологію якої є прихованою;
- розробити програмне забезпечення для тестування запропонованого методу маршрутизації транзакцій;
- провести аналіз безпеки фінансової топології та недоліків запропонованого методу маршрутизації транзакцій.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу:
- граф акторів мережі криптовалютних каналів з відкритою топологією;

- граф акторів мережі криптовалютних каналів з прихованою топологією;
- візуалізація взаємодії між початковою та кінцевою вершинами для встановлення параметрів транзакції;
- візуалізація етапу пошуку шляху в мережі криптовалютних каналів з прихованою топологією;
- візуалізація етапу побудови шляху в мережі криптовалютних каналів з прихованою топологією;
- діаграма класів розробленого модуля маршрутизації транзакцій для програмного забезпечення LND.

7. Орієнтовний перелік публікацій:
   - Тези доповіді "Адаптований спрощений протокол транзакційних каналів для криптовалютної системи Monero ".
   - Наукова стаття "Blind Payment Protocol for Payment Channel Networks".

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доц. каф. ПЗКС ФПМ		

9. Дата видачі завдання «11» жовтня 2017 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1.	Вивчення предметної області дослідження.	23.03.2018	
2.	Визначення структури магістерської дисертації; вивчення існуючих публікацій.	30.07.2018	
3.	Робота над першим розділом магістерської дисертації; проведення наукового дослідження.	03.09.2018	
4.	Проведення наукового дослідження; робота над другим розділом магістерської дисертації.	16.10.2018	
5.	Проведення наукового дослідження; підготовка матеріалів доповіді на конференцію ПМК-2018-2.	18.10.2018	
6.	Проведення наукового дослідження; робота над статтею за результатами наукового дослідження; розробка програмного забезпечення.	25.01.2019	
7.	Завершення роботи над основною частиною магістерської дисертації; підготовка ілюстративного матеріалу.	19.04.2019	
8.	Оформлення текстової і графічної частини магістерської дисертації	14.05.2019	

Студент _____ Ю.С. Жикін

Науковий керівник дисертації _____ М.В. Онай

РЕФЕРАТ

Актуальність теми. Мережі транзакційних каналів є найбільш ефективним з точки зору пропускної здатності застосування криптовалютної технології Bitcoin і є одним з найважливіших напрямків досліджень у галузі децентралізованих фінансових технологій. Основним процесом в такій мережі є маршрутизація транзакцій, оскільки саме вона визначає процес здійснення транзакцій. Єдина існуюча на даний момент реалізація такої мережі використовує фіксовану маршрутизацію, що зумовлює необхідність публічного доступу до топології мережі, яка в свою чергу містить інформацію про фінансові зв'язки між її учасниками. Для вирішення проблеми публічного доступу до топології мережі пропонується замінити метод детермінованої маршрутизації методом маршрутизації на основі тимчасових маршрутизаційних таблиць, що не використовує топологію мережі для побудови шляху. Такий метод маршрутизації дозволить використання множини приватних каналів в існуючій мережі для маршрутизації транзакцій без витоків даних про фінансові зв'язки між учасниками мережі.

Об'єкт дослідження – процес маршрутизації транзакції в мережі криптовалютних каналів в умовах відсутності даних про її топологію.

Предмет дослідження – методи, алгоритми та програмна реалізація маршрутизації транзакцій в мережах криптовалютних каналів.

Мета роботи – побудувати модель мережі транзакційних каналів з прихованою топологією та розробити метод маршрутизації транзакцій у ній.

Методи дослідження – моделювання мережі транзакційних каналів за допомогою теорії графів, композиція та адаптація існуючих методів

маршрутизації до предметної області, розробка програмного інтерфейсу маршрутизатора.

Наукова новизна:

1. Розроблено модель мережі криптовалютних каналів з прихованою топологією, яка відрізняється від існуючої моделі тим, що приховує інформацію про кількість вершин в мережі та зв'язки між ними.
2. Запропоновано метод маршрутизації транзакцій в мережі криптовалютних каналів, який відрізняється від існуючих тим, що не потребує доступу до інформації про топологію мережі, таким чином дозволяючи приховати цю інформацію і покращити характеристики приватності фінансових даних.

Практична цінність. Реалізацію запропонованого методу інтегровано в існуюче програмне забезпечення для роботи з мережею криптовалютних каналів Lightning, що дозволяє використовувати запропонований метод маршрутизації транзакцій в підмережі приватних криптовалютних каналів без витоків інформації про фінансові зв’язки між учасниками мережі.

Апробація роботи. Результати досліджень доповідалися та обговорювалися на XI науковій конференції магістрантів та аспірантів «Прикладна математика та комп’ютинг» ПМК-2018-2, яка відбулась у листопаді 2018 р., та опубліковані у збірнику тез доповідей за результатами конференції. За результатами досліджень була підготована та прийнята до друку в міжнародний журнал International Journal of Computer Network and Information Security (IJCNIS) наукова стаття на тему «Blind Payment Protocol for Payment Channel Networks».

Структура та обсяг роботи. Магістерська дисертація складається з вступу, чотирьох розділів, висновків та додатків.

У вступі зроблено огляд предметної області дослідження, сформульовано множину проблем, що розглядаються та обґрунтовано актуальність напрямку досліджень.

У першому розділі розглянуто компоненти криптовалютної системи Bitcoin, проблеми пропускної здатності та публічного доступу до транзакційних даних та вирішення проблеми пропускної здатності за допомогою зовнішніх транзакційних протоколів, що базуються на технології криптовалютних каналів. Описано принцип роботи криптовалютних каналів та ідею об’єднання множини таких каналів у мережу, що дозволить здійснювати багатокрокові транзакції вздовж певної послідовності каналів. Також описано компоненти протоколу багатокрокових транзакцій, поняття маршрутизації транзакцій у подібній мережі та загальні підходи до маршрутизації пакетів даних в комп’ютерних мережах.

У другому розділі побудовано модель мережі криптовалютних каналів на основі графа акторів та сформульовано транзакційний протокол мережі Lightning з точки зору цієї моделі. Окрім того, сформульовано поняття транзакційної, або фінансової топології та на основі графа акторів мережі продемонстровано встановлення фінансових зв’язків між фізичними сутностями за наявності асоціацій між ними та вершинами в мережі.

У третьому розділі запропоновано модель мережі криптовалютних каналів з прихованою топологією. В контексті такої мережі запропоновано метод маршрутизації транзакцій, що базується на заміні детермінованої маршрутизації використанням тимчасових маршрутних вказівок, що зберігаються в маршрутизаційних таблицях вершин на транзакційному шляху.

У четвертому розділі здійснено аналіз безпеки даних у мережі шляхом доповнення моделі мережі з прихованою топологією функцією існування

шляху, що відображає процес маршрутизації. Використовуючи цю функцію як модель зловмисника, сформульовано основну перевагу запропонованого методу – властивість приховування топології мережі, і продемонстровано, що така мережа зберігає дану властивість з точки зору такого зловмисника. Також подано аналіз недоліків запропонованого методу маршрутизації та запропоновано напрямки подальших досліджень.

У висновках наведено отримані результати роботи.

У додатках наведено копію презентації та лістинг фрагментів програмної реалізації.

Робота виконана на 74 сторінках, містить 8 рисунків, список використаних літературних джерел з 16 найменувань та 2 додатки.

Ключові слова: мережа криптовалютних каналів, криптовалюта, Bitcoin, маршрутизація.

ABSTRACT

Theme urgency. Payment channel networks are the most efficient applications of the Bitcoin cryptocurrency technology from the point of view of transaction throughput, and thus are one of the most important research areas in the sphere of decentralized financial technologies. The main process within such networks is the transaction routing, since it directly defines the process of making a transaction. The only active at the moment implementation of such networks uses source routing, which requires public access to the network topology, which in turn contains information about financial relations between the members of the network. In order to solve this problem, this thesis proposes to replace source routing with routing based on temporary routing tables, that doesn't use the network topology during route construction. This method allows to use the subset of private channels within the existing network to route transactions without leaking financial data.

Object of research – the process of transaction routing in the payment channel network with no access to its topology data.

Subject of research – methods, algorithms and software implementations of the transaction routing in the payment channel networks.

Research objective – develop a model of the payment channel network with the hidden topology and design a transaction routing method for such a network.

Research methods – graph-based network modelling, composition and adaptation of the existing routing methods to the given domain, applied programming interface design.

Scientific novelty:

1. Developed payment channel network model is different from the existing ones because it hides the information about the amount of nodes within the network as well as the connections between them.
2. Proposed transaction routing method is different from the existing one because it does not require access to the network topology data and thus allows to hide this information, consequently improving financial data privacy characteristics of the network

Practical value. Implementation of the proposed method is integrated in the existing software for interacting with the Lightning payment channel network, which allows to use the proposed transaction routing method in the private payment channel subnet without leaking information about financial relations between the members of the network.

Approbation. Results of the research have been presented and discussed at the 11th scientific conference for students and postgraduates “Applied mathematics and computing” PMK-2018-2 (Kyiv, November, 2018) as well formatted into a scientific paper «Blind Payment Protocol for Payment Channel Networks», accepted for publication in the International Journal of Computer Network and Information Security (IJCNIS, MECS-Press, Hong-Kong).

Structure and contents of the thesis. The master thesis consists ofo the introduction, four chapters, conclusions and appendixes.

Introduction contains an overview of research domain, important research problems as well as the argument for urgency of this particular research.

First chapter describes main components of the Bitcoin cryptocurrency system, the known problems of transaction throughput and data base transparency as well as the proposed solution to the transaction throughput problem using the external payment protocols that are based on the payment channel technology. It

describes the operation principles of payment channels and the idea to organize the set of such channels into a network, which allows to perform multi-hop transactions along a certain sequence of payment channels. It also describes the main components of the multi-hop payment protocol, the notion of transaction routing in such networks and gives details on general approaches to data packet routing in computer network.

Second chapter introduces the actor-graph-based model for the payment channel network and formulates the transaction protocol from the point of view of this model. It also formulates the notion of transactional or financial topology and using the actor graph of the example network, demonstrates how financial connections between external entities can be inferred when there are associations available between these entities and network nodes.

Third chapter proposes a model of the payment channel network with hidden topology. It also proposes a transaction routing method in the context of such network, which is based on replacement of source routing with temporary routing hints which are stored in the routing tables of the network nodes on the transaction route.

Fourth chapter performs an analysis of the data privacy in such networks by extending the network model with a path lookup function that represents the process of routing. Using this function as a model of an adversary, it formulates the main advantage of the proposed routing method – the property of hidden transaction topology and demonstrates that such network holds the property of hidden topology against the given adversary. It also contains the analysis of the disadvantages of the proposed method and lists potential areas of further research.

Conclusions describe the obtained results.

Appendices contain the copy of the slide deck and fragments of the implementation code base.

The thesis is presented on 74 pages, contains 8 diagrams, a list of 16 references and 2 appendices.

Keywords: payment channel network, cryptocurrency, Bitcoin, routing.

ЗМІСТ

ВСТУП

31 жовтня 2008 року невідомою особою під псевдонімом Сатоші Накамото було опубліковано статтю під назвою “Біткоїн: однорангова система електронної готівки” (англ. *Bitcoin: A Peer-to-Peer Electronic Cash System*) [1]. За допомогою однойменного програмного забезпечення 3 січня 2009 року цією ж особою було згенеровано перший блок розподіленої бази транзакцій, що за принципом роботи отримала назву *ланцюг блоків* (англ. *Block chain*). Спроби створення електронної валюти, що має характеристики готівки (анонімність, взаємозамінність тощо), в основному зосереджені навколо праць відомого криптографа і учасника шифропанк-руху Девіда Чома, велися ще з 80-х років минулого століття, але всі запропоновані до цього часу протоколи передбачали використання центрального органу, що здійснює випуск нових одиниць валюти. Біткоїн, багато в чому базуючись на цих ідеях, став першим протоколом електронної платіжної системи, що не потребує центральної точки контролю.

За 10 років розвитку криптовалютних технологій було здійснено багато досліджень, спрямованих на вирішення проблем оригінального протоколу, деякі з яких були сумісні з Біткоїн-мережею, і після детального розгляду спільнотою розробників були інтегровані в основну програмну реалізацію – Bitcoin Core Project, в той час як несумісні пропозиції або залишились на папері, або були трансформовані в програмне забезпечення для так званих “альтвалют” – альтернативних криптовалют. На даний момент існує близько тисячі різноманітних криптовалют з більш-менш релевантною ринковою капіталізацією, але лише невелика їх частина справді вирішує певну проблему на рівні протоколу і/або базується на важливих наукових результатах. Переважна більшість таких проектів є сучасними реалізаціями схеми “як швидко розбагатіти”.

Незважаючи на очевидні переваги протоколу Біткоїн порівняно не тільки з іншими пропозиціями електронної валюти, а й з всіма існуючими банківськими платіжними системами, ця технологія ще не набула того рівня застосування в різноманітних фінансових сферах, якого можна було б очікувати. В першу чергу це спричинено тим, що людство досить довго адаптується до кардинальних технологічних змін (варто згадати автомобілі та Інтернет-телефонію), але не менш важливим є й те, що відсутність центральної точки контролю не є безкоштовною – необхідність підтримки тисяч копій бази даних транзакцій та постійна загроза централізації через зменшення кількості таких копій є причиною однієї з найважливіших проблем даної галузі – пропускної здатності мережі. Ще одна проблема, яка має багато часткових вирішень, але при цьому є занадто складною для універсального вирішення, – це проблема порушення приватності фінансової інформації користувачів внаслідок публічного доступу до транзакційних даних, а отже необмежених можливостей застосування аналізу даних для корпоративного шпигунства чи цільової реклами.

У рамках даного дослідження розглядається одне з найбільш перспективних підходів до вирішення проблеми пропускної здатності – так звані транзакційні протоколи другого рівня – технології, що представляють транзакції як похідні дані від даних, якими оперує основна мережа. Такі протоколи використовують основну мережу як інструмент фіналізації транзакцій, тоді як основна маса фінансової інформації ніколи не потрапляє до базової криптовалютної мережі, економлячи на пропускній здатності. Єдина існуюча реалізація подібного протоколу використовує мережу учасників, зв’язаних між собою криптовалютними каналами [2], для здійснення багатокрокових транзакцій між каналами.

Побудова моделі такої мережі у вигляді графа акторів демонструє, що топологія мережі сама по собі є фінансовою інформацією, оскільки зв’язки в графі мережі відображають прямі грошові потоки між відповідними

їм сутностями. Для вирішення цієї проблеми у даному дослідженні пропонується реалізація модифікованого протоколу другого рівня на основі мережі прихованих криптовалютних каналів. Відсутність доступу до інформації про топологію мережі забезпечує приватність даних про фінансові зв’язки в межах мережі, але при цьому створює більше навантаження на мережу та сповільнює процес обміну транзакціями.

1. ОГЛЯД ІСНУЮЧИХ КРИПТОВАЛЮТНИХ ТЕХНОЛОГІЙ

1.1. Криптовалютна система Bitcoin

Всі сучасні криптовалютні протоколи базуються на технології, прототипом і основним зразком реалізації якої стало програмне забезпечення Bitcoin. Ця технологія полягає у агрегації даних мережі у блоки, кожен з яких окрім інформації про транзакції містить посилання на попередній блок, формуючи ланцюг блоків, який містить зв'язану історію всієї мережі починаючи з моменту запуску. Саме ця аналогія призвела до виникнення терміну *ланцюг блоків*.

Загальний принцип роботи криптовалют на основі технології ланцюгів блоків полягає в тому, що кожна транзакція повинна відповідати набору правил, які описують необхідні характеристики грошових одиниць і виконання яких легко перевірити вручну чи за допомогою певного програмного забезпечення. Цей набір правил називається консенсусом. Транзакції, які задовольняють вимогам консенсусу, потрапляють до блоків, які зв'язуються між собою за допомогою включення у кожен наступний блок ідентифікатора, отриманого шляхом обчислення надійної, з точки зору криптографічної безпеки, хеш-функції від даних попереднього блоку. Стійкість хеш-функції, що використовується, забезпечує цілісність даних у блоках, тоді як однозначність транзакційної історії досягається завдяки технології *PoW* (скор. *Proof of Work*, англ. *доказ виконаної роботи*). Ця технологія була вперше описана ще на початку 1990-х [3], і лягла в основу прототипу системи захисту електронної пошти від спаму *Hashcash* Адама Бека [4]. Алгоритм PoW системи Bitcoin, як і Hashcash, базується на частковій інверсії криптографічно-надійної хеш-функції і має наступні ключові характеристики:

1. Відносна (регульована) складність створення – пошук необхідного значення потребує тривалого перебору випадкових вхідних даних.

2. Простота перевірки валідності – одноразове обчислення заданої функції від вказаних вхідних даних.

Основна ідея захисту від повторного використання грошових одиниць у протоколі Bitcoin полягає в тому, що після формування і публікації транзакції система PoW усуває можливість її підміни засобом вміщення цієї транзакції у блок і виконання обчислювальної роботи такого обсягу, що робить переобчислення економічно невигідним.

У наступних пунктах вміщено детальний опис роботи компонентів системи, а саме структуру бази даних, технологію, що забезпечують стійкість і цілісність даних, транзакційну систему (так звані контракти) та основні проблеми даної технології, вирішення яких і є основною тематикою даного дослідження.

1.1.1. Ланцюг блоків

В основі криптовалютної технології лежить розподілена база даних, яку прийнято називати *ланцюгом блоків*. Структура цієї бази дозволяє забезпечити цілісність та стійкість до модифікації транзакційний даних.

Життєвий цикл транзакції виглядає наступним чином:

1. *Створення* – транзакція, що переводить кошти з однієї підмножини адрес на іншу, підписується власниками приватних ключів, які відповідають адресам вихідної підмножини і надсилається сусіднім учасникам мережу, які в свою чергу надсилають її далі, поширюючи таким чином по всій мережі.
2. *Басейн транзакцій* – кожна вершина в мережі, отримавши нову транзакцію, виконує її перевірку за правилами, визначеними протоколом консенсусу і розміщує її в спеціальній структурі в пам’яті для подальшої обробки. З цього моменту можна вважати, що транзакція здійснена, хоча й не підтверджена, оскільки видалення її з пам’яті кожної вершини в мережі потребує

значних ресурсів на зразок можливості дистанційного вимкнення пристроїв, на яких вона зберігається.

3. *Блок* – деякі вершини мережі виконують роботу по забезпеченню захисту транзакційний даних – так званий процес *майнингу* криптовалюти. За виконання цієї роботи передбачено винагороду, що є стимулом для все більшої кількості вершин брати у цьому участь. Для цього протокол Bitcoin визначає спеціальну складну математичну задачу, складність якої встановлюється поточним станом та параметрами мережі, яка формулюється наступним чином:

   > “Зібрати таку послідовність транзакцій з басейну транзакцій, що SHA256-хеш-сума від серіалізованого представлення цієї послідовності матиме d перших бітів зі значенням 0”,

   де d – число, що переобчислюється кожних N_D блоків, і обирається таким чином, щоб середній період між блоками p за наступні N_D блоків був якомога ближчим до періоду P, N_D – параметр мережі, що дорівнює 2016, P – параметр мережі, що дорівнює 600 секунд (10 хвилин). Таке формулювання дозволяє стверджувати, що будь яке рішення, яке його задовольняє, обчислювалось приблизно 10 хвилин на найбільш потужних обчислювальних пристроях, які приєднані до мережі. Більше того, будь-який приріст обчислювальної потужності в мережі призводить до зростання складності задачі рівно настільки, щоб зберегти 10-хвилинний інтервал. Послідовність транзакцій разом з певними метаданими утворює структуру, яка називається *блоком*, і над якою визначається умова описаної вище задачі. В якості винагороди за виконану роботу, вершина, яка знайшла розв’язок

задачі, може включити першою транзакцією в блок спеціальну транзакцію, яка генерує нові грошові одиниці, кількість яких визначена параметрами протоколу і зменшується у 2 рази кожні 210 тисяч блоків. Після того як було знайдено рішення задачі, блок з транзакцією надсилається сусіднім вершинам і поширюється по мережі. З цього моменту транзакція вважається підтвердженою.

4. *Зв'язування блоків* – одним з елементів даних, які входять до блоку, є хеш-сума попереднього блоку, через що базу даних з блокам і називають *ланцюгом блоків*. Криптографічні хеш-функції, однією з яких є хеш-функція SHA256, що використовується у протоколі Bitcoin, мають властивість, що називається *лавиновим ефектом*, і полягає у тому, що зміна одного біта у вхідних даних спричиняє зміну кожного біта у вихідному рядку з ймовірністю 0.5. Оскільки хеш-сума блока B_i включена в блок B_{i+1}, зміна навіть одного біта в блоку B_i спричинить зміну половини бітів його хеш-суми, а отже інвалідує хеш-суму блока B_{i+1}, що означає, що для модифікації послідовності транзакцій в одному блоці необхідно переобчислити задачу для нього і всіх наступних блоків $B_j, j = i+1, i+2, ..., k$ після нього, де k – найбільш новий блок у базі даних. Ця задача є значно складнішою, оскільки за визначенням процесу Proof-of-Work період у 10 хвилин є однаковим як для обчислення задачі для блока B_j, так і для блока B_{k+1}, тому єдиний спосіб вдалого проведення атаки по заміні блока в базі даних – взяти під контроль переважну більшість обчислювальної потужності всієї мережі – занадто складний навіть для сутностей рівня державних апаратів. При цьому максимальний вплив такої атаки на мережі – відміна транзакцій що містяться у блоках $B_i, B_{i+1}, ..., B_k$ (усі надіслані кошти всього лише повернуться їхнім попереднім власникам).

За допомогою оцінки обчислювальних ресурсів мережі можна визначити, що якщо транзакція знаходиться у блоці B_i, а поточний блок в базі – B_k, то за умови, що $k - i \geq 6$, видалення транзакції потребує ресурсів, що перевищують ресурси будь-якої держави на планеті, що дозволяє стверджувати, що після 6 блоків транзакція є остаточно підтвердженою.

1.1.2. Скриптова система

Тоді як ланцюг блоків та система Proof-of-Work забезпечують цілісність та стійкість бази даних транзакцій, друга складова протоколу консенсусу Bitcoin визначає, яким чином встановлюється право власності над конкретними коштами у базі даних. Ця частина протоколу будується на основі спеціальної структури транзакції і полягає у тому, що кожна транзакція є атомарною модифікацією всього стану бази, яка в свою чергу містить повну історію таких модифікацій.

Ключовим поняттям протоколу Bitcoin є так званий "невитрачений вихід транзакції" (англ. *Unspent transaction output, UTxO*), який представляє собою одночасно певну кількість абстрактних одиниць цінності та право власності на них. Для визначення цього поняття необхідно спочатку розглянути структуру транзакції.

Кожна транзакція складається з двох основних компонентів – множин так званих *виходів* (англ. *Transaction output, TxOut*) та *входів* (англ. *Transaction input, TxIn*):

1. *Вихід транзакції* складається з цілочисельної величини, що означає кількість абстрактних одиниць цінності ("1 біткоїн" складається з 10^8 атомарних одиниць, які стало прийнято називато "сатоші" на честь автора оригінальної ідеї), та спеціальної криптографічної задачі, розв'язок якої доводить право власності на вказану кількість коштів.

2. *Вхід транзакції* складається з посилання на наявний в базі даних вихід транзакції та розв'язку відповідної криптографічної задачі. Якщо розв'язок задачі відповідає умові з вказаного виходу, цей вихід стає входом у даній транзакції, а сама транзакція вважається дійсною.

Використовуючи дані поняття, *невитрачений вихід транзакції* можна визначити як такий вихід, який не є входом жодної транзакції.

Отже, будь-яка транзакція є операцією над множиною невитрачених виходів транзакцій, яка видаляє певну підмножину елементів цієї множини та створює нові елементи. При цьому всі кошти, що існують в мережі, створюються в так званій базовій (англ. *Coinbase*) транзакції, яку, як було зазначено у пункті 1.1.1, має право додати в перелік транзакцій блока вершина мережі, що знайшла розв'язок задачі для даного блока. Ця транзакція має 0 входів та 1 вихід, який і генерує нові грошові одиниці.

Підсумовуючи, можна сказати, що факт володіння певними коштами у мережі Bitcoin означає, що сутність-власник знає розв'язки задач якоїсь підмножини невикористаних виходів транзакцій.

Найцікавішим компонентом протоколу Bitcoin є те, що як самі задачі, які накладають умови власності над коштами, так і їх розв'язки, є довільними програмами, поданими спеціальною неповною за Тюрінгом (мається на увазі відсутність мовних засобів для опису циклів) мовою програмування, яку називають *Біткоїн-скриптом*, або просто *Скриптом* (англ. *Script*). Процес перевірки умов власності має дуже просту семантику – програма-умова з виходу транзакції об'єднується з програмою-розв'язком з відповідного входу, завантажується в інтерпретатор скрипта та виконується. Якщо результатом виконання об'єднаної програми є булеве значення *True*, транзакція є дійсною.

Використання спеціальної мови для опису задач-умов власності та їх розв'язків має надзвичайно важливий наслідок як для протоколу так і для

всіх сфер його застосування: Bitcoin – це монетарна система, одиниці якої можна програмувати.

1.1.3. Пропускна здатність

Однією з основних проблем всіх сучасних криптовалютних протоколів загалом, і протоколу Bitcoin зокрема, є пропускна здатність. Оскільки для генерації нового блоку в умовах PoW необхідний заданий параметрами протоколу проміжок часу, кількість оброблених транзакцій за одиницю часу обмежена розміром блока та тривалістю періоду між блоками. Так, оскільки період між блоками в мережі Біткоїн становить $\tau = 10$ хвилин, розмір блока зафіксовано як $B = 1$ мегабайт, а середній розмір транзакції $S = 600$ байт, пропускна здатність мережі становить

$$B/\tau S \approx (10^6 \text{ Б})/(600 \text{ Б/т.} * 10 * 60 \text{ с}) \approx 3 \text{ транзакції/с.}$$

Для порівняння з традиційними, централізованими платіжними системами, середня пропускна здатність системи VISA становить 4000 транзакцій/с, а заявлена максимальна пропускна здатність – близько 60 000 транзакцій/с, на кілька порядків вищі за можливості Біткоїн-мережі.

Було запропоновано декілька тривільних рішень цієї проблеми, але усі вони є субоптимальними, оскільки вони або не дають результатів на тривалих проміжках часу або конфліктують з деякими важливими характеристиками мережі:

1. *Оптимізація структури транзакцій* – пошук оптимального розміщення компонентів транзакцій з метою зменшення їх середнього розміру; хоча є дуже важливою галуззю досліджень, дає *сублінійний* приріст пропускної здатності, оскільки розмір транзакції може зменшуватись лише на певний коефіцієнт $k < 1$ і є обмеженим знизу мінімільним розміром ідентифікаційних даних (хеш-суми) транзакції.

2. *Збільшення розміру блока* – більший блок вміщує більше транзакцій, але збільшення розміру блока на коефіцієнт $k > 1$ збільшує приріст розміру бази даних за одиницю часу на той же коефіцієнт, що призводить до централізації мережі, оскільки з часом все менша кількість учасників можуть дозволити собі зберігати повну базу даних і змушені використовути *збирання сміття*, яке видаляє частину "старих" блоків і унеможливлює повну перевірку відповідно до консенсусу; зменшення кількості учасників з повними копіями даних підвищує ймовірність успіху атак на цілісність даних; окрім того, дане рішення є тимчасовим, оскільки дає *лінійний* приріст пропускної здатності.
3. *Зменшення періоду між блоками* – менший період між блоками означає більшу кількість блоків, а отже і транзакцій, за одиницю часу, що призводить до тих же проблем що й збільшення розміру блока, описане вище, але при цьому має ще й суттєвий недолік, а саме зменшення стабільності мережі, оскільки чим коротший проміжок часу між блоками, тим складніше досягнути повної синхронізації мережі перед появою наступного, що уможливлює атаки розділення мережі.

Слід зазначити, що оптимізація структури транзакції та блока є дуже цікавою галуззю для досліджень. Основним напрямком цих дослідження є винесення менш важливих чи великих за розміром компонентів у відокремлені структури, що за необхідності можуть бути відкинуті. Так, однією з найважливіших останніх змін в протоколі Біткоїн була реалізація BIP-141 [5] – пропозиції щодо винесення порівняно великих за розміром криптографічних підписів з кожної транзакції у окрему структуру, що отримала назву *SegWit* (англ. *Segregated Witness* – "відокремлений свідок"). Ця модифікація протоколу є першим кроком до реалізації більш загального механізму фіксації в блоці довільних даних без вміщення самих

даних безпосередньо в тілі блока. Одним з найцікавіших застосувань такого механізму є пропозиція BIP-114 [6] щодо вміщення в транзакції лише кореня MAST – “меркелізованого” синтаксичного дерева скрипта транзакції і винесення самих скриптів у окрему структуру, що дозволить зняти обмеження з їхнього розміру і таким чином надати користувачам можливість створення довільних транзакційних контрактів і повністю розкрити одну з найменш відомих, але при цьому найбільш цікавих властивостей криптовалюти Біткоїн – програмовність грошових одиниць.

Втім, подібні дослідження спрямовані більше на узагальнення протоколу і на створення нових сфер застосування, ніж на підвищення пропускної здатності, оскільки ніякі оптимізації структури блоків та транзакцій на дозволять Біткоїн мережі наблизитись до пропускної здатності тієї ж системи VISA. Більш радикальний підхід полягає у створення нового децентралізованого транзакційного протоколу, що використовує Біткоїн лише як засіб остаточної фіксації транзакцій, зберігаючи проміжний стан поза межами ланцюга блоків. Більш детально цей підхід розглянуто в підрозділі 1.2.

1.1.4. Транзакційна прозорість

Друга серйозна проблема протоколу Біткоїн є наслідком того, що верифікація процес верифікації блоків передбачає виконання скриптів у кожній транзакції, і тому транзакційні дані, накопичені протягом всього періоду функціонування мережі зберігаються в ланцюгу блоків у відкритому вигляді. Хоча (в кращому випадку) сутності-учасники транзакцій приховуються за одноразовими псевдонімами – адресами, кореляційний аналіз даних ланцюга блоків дозволяє з певною ймовірністю встановлювати зв’язок між окремими адресами, що належать одній і тій же сутності. Подібні статистичних аналізи порушують характеристику анонімності грошової системи і унеможливлюють використання

криптовалюти в умовах тоталітарних режимів чи для здійснення корпоративних закупівель, виплати заробітньої плати, тощо. Більш серйозним наслідком прозорості транзакційних даних є те, що кожна грошова одиниця має певну історію – послідовність попередніх власників. Так, деякі компанії можуть відмовитись приймати оплату криптовалюто, яка історично пов'язана з адресою, що з певною імовірністю належить продавцеві нелегальних товарів, що порушує характеристику взаємозамінності грошових одиниць. Таким чином Біткоїн, всупереч назві статті, у якій він був описаний ("Децентралізована система електронної готівки"), не має двох ключових характеристик готівки – анонімності та взаємозамінності.

Розглядаючи абстрактну модель транзакції, а саме кортеж $(K_{sender}, K_{receiver}, A)$, де K_{sender} – ключ адресанта транзакції, $K_{receiver}$ – ключ адресата транзакції, A – кількість коштів, що надсилаються, можна неформально визначити систему анонімізації транзакцій як набір засобів, що дозволяють приховати одну або більше складових транзакції. На даний момент можна виділити два основних підходи до анонімізації транзакцій, реалізовані у криптовалютних системах Zcash та Monero відповідно: система zero-knowledge-доведення валідності транзакцій та система обфускації їх складових. Ключовою відмінністю між даними рішеннями є те, що система zkSNARK [7], що використовується у системі Zcash, базується на математичному апараті, який на даний момент ще не має достатньої кількості емпіричних доказів криптографічної безпеки, тоді як система обфускації транзакцій Monero базується на простих криптографічних примітивах, що використовуються у найрізноманітніших галузях інформаційних технологій протягом останніх 30-ти років.

Проблема застосування засобів обфускації транзакційних даних в криптовалютному протоколі Bitcoin полягає в тому, що такі зміни значно ускладнюють реалізацію та тестування системи, тому, як

уже було зазначено раніше, спільнота розробників протоколу не розглядає такі зміни як можливі найближчим часом. Натомість розробка протоколів другого рівня дозволяє винести основні фінансові взаємодії за межі публічно доступних даних, тому велика кількість досліджень, пов'язаних з приватністю транзакційних даних, і дане дослідження в тому числі, зосереджуються на опосередкованих засобах анонімізації фінансової інформації, таких як гігієнічне використання криптографічних компонентів та мінімізація поширення метаданих транзакцій. Зокрема, дане дослідження розглядає потенційну можливість приховування фактів існування фінансових зв'язків між учасниками транзакційних мереж засобом приховання інформації про всі зв'язки в мережі.

1.2. Мережа транзакційних каналів Lightning

Однією з найцікавіших пропозицій щодо підвищення пропускної здатності Біткоїн-мережі є ідея винесення основної маси транзакцій в окрему мережу, глобальний стан якої розподілений між вершинами мережі і не потребує перевірки та зберігання в усіх учасників. Кожна сутність у такій транзакційній системі зберігає лише власний стан, при цьому сам ланцюг блоків використовується для фіналізації накопичених транзакційних станів.

Подібна платіжна система, яку часто називають "мережею другого рівня" (з англ. *Second layer network*, або просто *Layer2*), складається з множини транзакційних каналів між окремими сутностями-учасниками мережі, та засобів передачі коштів між каналами. Таким чином, основна маса транзакцій криптовалютної мережі відбуває безпосередньо між двома учасниками мережі другого рівня, при цьому деякі з транзакцій є насправді ланками шляху передачі коштів між кількома каналами. Регулючи глобальний середній період фіналізації транзакцій, можна

досягнути пропускної здатності, що обмежена лише швидкістю передачі даних між двома вершинами, оскільки про факт надсилання коштів достатньо знати лише цим вершинам.

В даному підрозділі детально розглядаються конструкції транзакційних каналів та засоби передачі коштів між окремими такими каналами.

1.2.1. Транзакційний канал

Ключовою ідеєю, що лежить в основі поняття транзакційного каналу, є той факт, що необхідним та достатнім доказом передачі коштів в Біткоїн-протоколі є підписана адресантом транзакція, а отже для передачі коштів достатньо підписати сформований транзакційний документ і передати його адресату, який, в свою чергу може опублікувати його в будь-який момент часу, остаточно закріпивши за собою власність над коштами. Таким чином дві сутності можуть обмінюватись неопублікованими транзакціями, підтримуючи кінцевий баланс на необхідному для їхнього контракту рівні. Втім, тривіальна реалізація такого інструменту повертає нас до проблеми повторного використання, яку криптовалютні системи власне і намагаються вирішити в першу чергу: якщо адресант опублікує транзакцію, яка надсилає ті ж кошти на іншу адресу, то збережена транзакція стає недійсною і повторне використання коштів завершується успіхом. Через це для створення безпечних транзакційних каналів необхідно перш за все сформувати спеціальний “рахунок”, який зберігатиме всі кошти обох сторін, а також надати набір криптовалютних примітивів, що дозволяють формулювати складніші форми контрактів, такі, як, наприклад:

> “Дані кошти можна перевести на іншу адресу не раніше ніж за N блоків після блоку, в якому знаходиться дана транзакція.”

Загальний життєвий цикл транзакційного каналу складається з наступних кроків:

1. Ініціалізація – дві сторони створюють багатопідписний гаманець, який містить грошові кошти s_1 та s_2 сторін A, та B відповідно (загальна кількість коштів у гаманці $s = s_1 + s_2$). Окрім цього дві сторони створюють нову транзакцію, яка повертає сторонам каналу вкладені ними кошти з багатопідписного гаманця, створеного на етапі ініціалізації. У цей момент канал вважається активним, і його закриття поверне рахунки двох сторін до початкового стану, тому вважається, що сторони мають ту ж кількість коштів, що й до відкриття каналу.
2. Активація – транзакція, що створює багатопідписний гаманець, публікується в мережі, і з моменту підтвердження утворює новий транзакційний канал. Транзакція, що повертає сторонам кошти, не публікується в мережі (оскільки її публікація одразу ж закриє канал), і формує початковий стан каналу.
3. Оплата – основна складова життєвого циклу каналу, яка повторюється довільну кількість разів, і полягає у виконанні протоколу, що створює новий стан каналу, одночасно інвалідуючи попередній стан – публікація "старого", недійсного стану призводить до виконання протоколу арбітрації, який визначає порушника і накладає певні санкції. При цьому новий стан канал складається з транзакції, яка надсилає сторонам кошти $s = s'_1 + s'_2$, де $s'_1 = s_1 - p$, $s'_2 = s_2 + p$, що відображає надсилання суми p стороною A стороні B.
4. Закриття – будь-яка з сторін публікує в криптовалютній мережі транзакцію, що відображає поточний стан каналу, таким чином фіналізуючи кінцеві рахунки і остаточно переводячи відповідні суми обом сторонам.

5. Арбітрація – у випадку, якщо опублікована під час закриття транзакція не є остаточним станом каналу, публікація версії остаточного стану з точки зору іншої сторони каналу, ініціює протокол арбітрації, який визначає остаточний стан каналу і накладає санкції на сторону, що порушила контракт.

1.2.2. Непрямі транзакції

Одним з найбільш амбітних та масштабних проектів на основі мережі Bitcoin є Lightning Network – мережа, що складається з сутностей, пов'язаних між собою транзакційними каналами, що дозволяє створювати мікротранзакції, здійснювати пошук послідовності каналів, що виконує функцію шляху передачі коштів та здійснювати позамережеву передачу криптовалютних одиниць "вздовж" такого шляху. Таким чином група сутностей може обмінюватись коштами з пропускною здатністю, що обмежена лише швидкістю Інтернет-з'єднання, без використання ресурсів криптовалютної мережі. Основним компонентом проекту є протокол пересилання транзакцій між каналами, для реалізації якого у мову транзакційного скриптування системи Bitcoin було впроваджено додаткові операції.

Основним інструментом для пересилання транзакцій між каналами є протокол *HTLC* (англ. *Hash Timelock Contract*). Цей тип Bitcoin-контракту полягає в тому, що одна з сторін може отримати кошти, що "замкнуті" у даній транзакції, надавши доказ того, що їй відоме спеціальне секретне значення. Неформально такий контракт можна описати наступним чином:

> "Протягом періоду часу t дані кошти можна перевести на іншу адресу за умови, що нова транзакція буде супроводжуватись спеціальним значенням x таке, що $HASH(x) = p$, де p задано в контракті; після завершення терміну t, кошти повернуться власнику."

Використовуючи дану конструкцію, група учасників може побудувати послідовність HTLC-транзакцій відносно однієї й тієї ж пари значень $(x, p = HASH(x))$, так, що починаючи від отримувача і вздовж утвореного ланцюжка контрактів, кожен учасник отримує значення x від попереднього учасника, і за допомогою цього значення може отримати кошти у наступного в обмін на значення x. Пересилання транзакції за допомогою HTLC відбувається наступним чином (тут покупець – сторона, що надсилає кошти, продавець – сторона, що їх отримує):

1. Продавець генерує випадкове значення x з достатньою кількістю ентропії, обчислює значення $p = HASH(x)$ і передає його покупцеві.
2. Покупець здійснює пошук шляху – послідовності учасників мережі – до продавця і створює HTLC-контракт відносно значення p з першим учасником знайденої послідовності.
3. Кожен посередник платежу створює новий HTLC-контракт з наступним членом послідовності відносно того ж значення p, фактично утворюючи послідовність декрементів/інкрементів балансів протилежних кінців транзакційних каналів вздовж шляху.
4. Продавець передає значення x "найближчому" до нього учаснику платіжного шляху, виконуючи свою сторону HTLC-контракту та отримуючи відповідну кількість грошових одиниць.
5. Оскільки всі HTLC-контракти стосуються одного й того ж значення p, кожен учасник шляху, отримавши від наступного значення x, може виконати свою сторону контракту з попереднім учасником, передавши значення x йому, таким чином створюючи ефект "перетікання" грошових одиниць з одного каналу в інший вздовж зазначеного шляху.
6. У випадку відмови від кооперації деяких учасників шляху, поширення платежу може бути скасована за допомогою другої

умови контракту – після завершення періоду часу t контракт скасовується, і кошти повертаються до власника.

Більш детально реалізацію каналів та прокотокол міжканального пересилання транзакцій описано у оригінальній статті [2] Джозефа Пуна та Таддеуша Драйджі від 2016 року.

1.2.3. Проблема маршрутизації

Оскільки технологія мережі транзакційних каналів є новою навіть за мірками галузі криптовалютних технологій (перша публікація датується 2016-м роком), поточна специфікація виділяє досить багато відкритих питань для дослідження. Більшість цих питань пов'язані з розширенням функціональності технології (наприклад *поповнення каналів* – можливість збільшити кількість коштів, замкнутих в межах каналу уникаючи закриття/відкриття, чи *складені транзакції* – транзакції, що складаються з кількох менших транзакцій і завершуються успішно лише за умови, що всі транзакції-компоненти були успішними), але деякі з них стосуються базових компонентів протоколу та їхніх обмежень (наприклад, сам *протокол транзакційного каналу*).

Дане дослідження розглядає один з базових протоколів мережі транзакційних каналів, а саме протокол маршрутизації транзакцій в мережі, і застосовує графову модель взаємодії акторів мережі для опису існуючого протоколу маршрутизації, формулювання його недоліків, а також для опису розробленого в рамках дослідження модифікованого протоколу маршрутизація та обгрунтування його переваг.

Під маршрутизацією транзакцій мається на увазі протокол здійснення транзакції між учасниками мережі, які не є зв'язані транзакційними каналами безпосередньо. Неформально описаний у пункті 1.2.2, цей протокол має формальне визначення, аналогічне визначенню більшості протоколів TCP/IP – дані, якими оперують учасники протоколу,

організовані в спеціальну структуру, що називається *пакетом*, а сукупність алгоритмів, які виконують учасники протоколу, та взаємозв'язок між ними і називаються процесом *маршрутизації*.

У наступному підрозділі розглядається загальна теорія маршрутизації пакетів в комп'ютерних мережах та окремі підходи до маршрутизації, які тою чи іншою мірою стосуються даного дослідження.

1.3. Загальне поняття маршрутизації пакетів

Маршрутизацією називають процес визначення маршруту інформації в мережі. Проблема маршрутизації детально вивчається з ще з моменту появи телефонних мереж, тоді як найбільший обсяг досліджень припадає на мережі з комутацією пакетів, прикладом яких є Інтернет. В межах даного дослідження розглядаються три категорії підходів до маршрутизації: *маршрутизація від джерела* та *цибулева маршрутизація*, які безпосередньо використовуються в існуючій реалізації мережі транзакційних каналів, та *динамічна (таблична) маршрутизація*, яка й розглядається в даному дослідженні як альтернатива, що забезпечує вирішення сформульованої проблеми.

1.3.1. Маршрутизація від джерела

Маршрутизація від джерела (англ. *Source routing*, також відома як *адресація шляху* (англ. *path addressing*) – спосіб маршрутизації пакетів у комп'ютерних мережах, що дозволяє адресанту пакета частково чи повністю задати шлях, по якому пакет подорожуватиме мережею. Різні варіації даного способу маршрутизації підтримуються багатьма мережевими технологіями, зокрема один з найважливіших комунікаційних протоколів сучасності, IP, дозволяє його використання через опції заголовка SSRR та LSRR, хоча остання опція часто блокується в мережі

через проблеми безпеки, до яких вона може призвести (мається на увазі можливість зловмисника підмінити власну IP-адресу і при цьому продовжити отримувати пакети-відповіді). Втім, історично, ці опції використовувались для моніторингу роботи мережі та маршрутизаторів на певних її ділянках.

Алгоритм маршрутизації від джерела дуже простий – кожна вершина, що бере участь у маршрутизації, отримує пакет, дістає з його заголовка вказану послідовність адрес (маршрут), знаходить свою власну адресу в цій послідовності, і отримує адресу наступної вершини, беручи наступний елемент послідовності, після чого виконує надсилання пакета за отриманою адресою замість кінцевої адреси, вказаної в пакеті. Цікавим є те, що за умови використання стандартної табличної маршрутизації (див. пункт 1.3.2) на етапі надсилання пакета наступній вершині на шляху, алгоритм працюватиме навіть якщо шлях вказано лише частково.

Протокол маршрутизації мережі Lightning базується на методі маршрутизації від джерела, але використовує технологію цибулевої маршрутизації (див. пункт 1.3.3) для забезпечення основних характеристик транзакційної безпеки.

1.3.2. Динамічна маршрутизація

Динамічна маршрутизація – один з найпоширеніших типів маршрутизації, є основним режимом роботи протолу IP [8]. Принцип роботи цього методу маршрутизації полягає в тому, що кожна вершина-маршрутизатор в мережі має таблицю, яка називається *таблицею маршрутизації*, і містить множину маршрутних вказівок (P_{target}, A_{next}), де P_{target} – префікс адреси вершини-отримувача пакету, а A_{next} – адреса наступної вершини-маршрутизатора на шляху. Варто зазначити, що остаточний шлях в даному випадку невідомий аж до моменту доставки пакета, оскільки таблиці маршрутизації можуть мінятись протягом часу

(для цього передбачено спеціальні протоколи синхронізації маршрутної інформації), через що цей тип маршрутизації і називають динамічним.

1.3.3. Цибулева маршрутизація

Цибулева маршрутизація – технологія анонімізації трафіку в мережі, розроблена в середині 90-х років Дослідницькою лабораторією Військово-морського флоту США (NRL) та Агентством передових оборонних дослідницьких проектів США (DARPA) для військових цілей. На початку 2000-х років група дослідників розробила основну реалізацію протоколу, яка отримала назву *Tor* (англ. *The Onion Routing – Цибулева маршрутизація*). Ця назва з часом набула більш широкого застосування як назва класу протоколів.

Основна ідея даної технології полягає в тому, що всі пакети мають форму “цибулини”, де основна інформація захована у “шари” несиметричного шифрування різними ключами, і лише останній учасник процесу маршрутизації (отримувач пакету) може “зняти” останній шар.

Для здійснення маршрутизації надсилач пакета використовує маршрутизацію від джерела, будуючи шлях в пакета в мережі. Для цього він обирає множину вершин зі списку що надається спеціальною вершиною, яка називається *директорією* і по суті є дистрибютором інформації про топологію мережі. Отримавши публічні ключі цих вершин, надсилач пакету шифрує дані цими ключами, починаючи від останнього і додаючи на кожному кроці інформацію про наступну вершину в списку, після чого передає пакет першій вершині з обраного списку. Таким чином кожен учасник маршрутизації може зняти з пакету лише один шар шифрування (той, що відповідає його власному ключу), отримати додаткові маршрутизаційні дані і передати пакет далі.

Мережу Tor часто помилково називають децентралізованою, але оскільки інформація про топологію контролюється

вершинами-директоріями, ця класифікація є некоректною. Необхідність централізації маршрутизаційних даних є одним з основних недоліків цієї мережі, оскільки створює проблему “вразливої точки” – одночасна атака на всі вершини-директорії виводить мережу з дії, і, окрім того, вершина-директорія, що порушує правила, ставить під загрозу всіх користувачів мережі. В той же час – це один з ефективних і простих способів реалізації протоколу розгортання мережі (англ. *discovery*), особливо враховуючи загальну складність задачі децентралізованої репутації.

1.4. Висновки

В даному розділі розглянуто принципи роботи криптовалютних систем та дві основні проблеми, що є наслідками архітектури таких систем – проблему низької пропускної здатності, спричиненої необхідністю зберігання десятків тисяч копій бази даних транзакцій на комп’ютер добровільних учасників мережі для підтримки децентралізації, а також проблему відкритого доступу до транзакційних даних, спричинену необхідністю кожного учасника в мережі мати змогу прослідкувати всю історію своїх коштів до моменту їх генерації.

Розглянуто запропоноване вирішення проблеми пропускної здатності криптовалютної системи Bitcoin на основі технології транзакційних каналів – спеціального криптовалютного контракту, що дозволяє двом його сторонам обмінюватись транзакціями без публікації їх в мережі. Ідея рішення полягає в тому, що множину таких транзакційних каналів можна об’єднати у мережу за допомогою додаткового криптовалютного контракту, що створює умовно-відтерміновані транзакції і таким чином дозволяє учаснику такої мережі безпечно переводити кошти з одного каналу в інший.

Описано проблему, на якій зосереджене дане дослідження, а саме той факт, що використання фіксованих шляхів маршрутизації потребує доступу до інформації про топологію мережі, яка безпосередньо відображає фінансові зв’язки між вершинами мережі, і таким чином становить потенційну небезпеку приватності фінансових даних учасників мережі. Також розглянуто загальні підходи до маршрутизації пакетів даних у комп’ютерних мережах.

2. МОДЕЛЬ МЕРЕЖІ ТРАНЗАКЦІЙНИХ КАНАЛІВ

2.1. Граф мережі

Мережу транзакційних каналів можна розглядати як граф $(\mathcal{N}, \mathcal{C})$, де $\mathcal{N}$ – множина сутностей, що взаємодіють одна з одною згідно з базовим протоколом криптовалютної мережі, а $\mathcal{C}$ – множина транзакційних каналів.

Для кожної транзакції, що проходить в мережі транзакційних каналів, існує певна підмножина вершин мережі, які беруть безпосередню участь в процесі її виконання. Разом з певним встановленим над нею порядком, ця підмножина називається *шляхом транзакції*, $\mathcal{P} \subset \mathcal{N}$.

2.1.1. Класи акторів мережі

Виділення в мережі транзакційних каналів шляху транзакції розбиває множину вершин мережі на наступні класи:

1. *Вихідні вершини* $\mathcal{S}$, або *вершини-надсилачі* (англ. *Senders*) – множина початкових вершин шляху транзакці, що виконують роль формування та ініціювання транзакції. Оскільки технології, що дозволяють розподілені транзакції, знаходяться на етапі розробки, для простоти опису в даному дослідженні вважається, що множина вихідних вершин складається з однієї вершини $\mathcal{S} = \{S\}$.
2. *Кінцеві вершини* $\mathcal{R}$, або *вершині-отримувачі* (англ. *Receiver*) – остання вершина шляху транзакції, що виконує роль створення запиту на транзакцію та ініціювання поширення HTLC. Аналогічно до множини вихідних вершин, в рамках даного дослідження розглядається одноелементна множина кінцевих вершин $\mathcal{R} = \{R\}$.
3. *Вершини-посередники* $\mathcal{I}$ (англ. *Intermediaries*) – всі вершини на шляху транзакції, окрім вихідної та кінцевої, $\mathcal{I} = \mathcal{P} \setminus \{S, R\} = \{I_1, I_2, ..., I_n\}$. Під час виконання транзакційного

протоколу виконують переведення коштів з каналу (I_{i-1}, I_i) в канал (I_i, I_{i+1}), використовуючи контракт HTLC (див.

4. *Зовнішні вершини* $\mathcal{E}$ (англ. *External nodes*) – всі інші вершини мережі, $\mathcal{E} = \mathcal{N} \setminus \mathcal{P} = \{E_1, E_2, ...\}$, і за консервативною моделлю безпеки вважається, що вони можуть "підслуховувати" інформацію про дану транзакцію.

Приклад графа акторів мережі та розподіл множини вершин на класи акторів відносно визначеного транзакційного шляху подано на рис. 2.1.

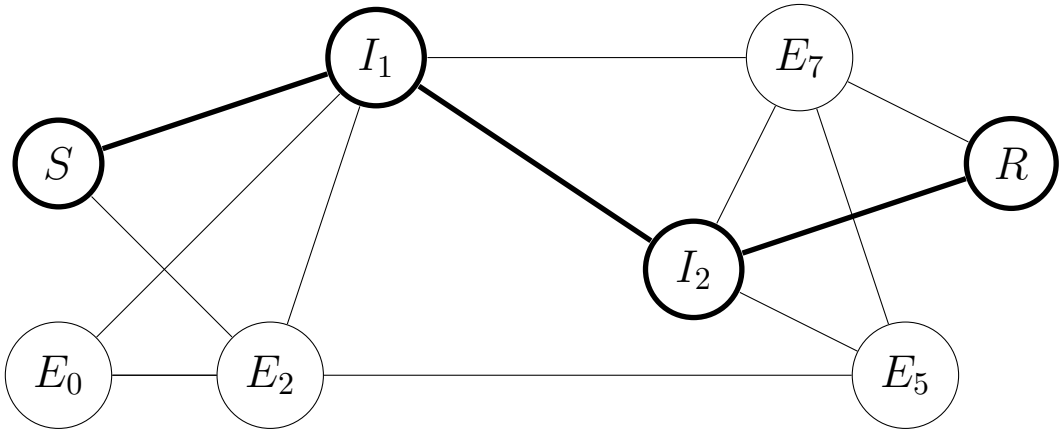


Рис. 2.1. Граф акторів мережі

Оскільки як вершини, так і зв'язки в графі акторів мережі розглядаються в контексті, що відображає інформацію про відношення *Надсилач-Отримувач* та фінансові потоки між ними, у даному дослідженні пропонується називати сукупність даних про існуючі вершини, та зв'язки між ними *транзакційною або фінансовою топологією*. Слід зазначити, що ця інформація отримується безпосередньо з даних про форму графа мережі, без застосування додаткового аналізу.

2.1.2. Протоколи взаємодії між акторами мережі

В рамках даної моделі розглядається два набори розподілених алгоритмів – протоколи взаємодії вершин в мережі:

1. *Протокол транзакційного каналу* – протокол, що формує динамічним зв'язок між двома сусідніми вершинами в мережі і з точки зору моделі представлений ребром в графі мережі, а з точки зору реалізації – контрактом транзакції, що відкриває цей канал в основній мережі Bitcoin.
2. *Транзакційний протокол* – протокол, що об'єднує ланцюжок транзакційних каналів в мережі в транзакційний шлях для здійснення окремої транзакції. Вибір загальної назви пояснюється тим, що всі транзакції в мережі здійснюються за цим протоколом, навіть якщо множина вершин-посередників $\mathcal{I}$ – порожня.

Протокол транзакційного каналу описано у пункті 1.2.1.

Транзакційний протокол складається з трьох розподілених алгоритмів:

1. *Request* – взаємодія між вихідною та кінцевою вершинами на транзакційному шляху, що встановлює параметри транзакції та дозволяє початковій та кінцевій вершинам обмінятись необхідними криптографічними матеріалами.
2. *Send* – взаємодія між вихідною вершиною та першою вершиною на транзакційному шляху, якій передує пошук шляху в графі мережі.
3. *Forward* – ключовий компонент транзакційного протоколу, послідовна взаємодія згідно з протоколом транзакційного каналу між вершинами на встановленому транзакційному шляху вздовж, що базується на так званих контрактах *HTLC* (див. пункт 1.2.2) і полягає у тому, що вершина-посередник I_i, що має канали (I_{i-1}, I_i) та (I_i, I_{i+1}), може здійснити "перетягування" коштів з одного каналу в інший.

2.2. Процес маршрутизації

Подану в попередньому підрозділі модель графа акторів мережі можна використати для опису процесу маршрутизації транзакцій з використанням оригінального методу маршрутизації “від джерела”.

Схема маршрутизації транзакцій Lightning базується на конструкції Sphinx [9], доповненої інформацією, необхідною на кожному кроці пересилання пакетів.

Вершини-посередники при пересиланні можуть перевірити цілісність пакета і дізнатись про те, яка вершина є наступною на транзакційному шляху. Інформація про будь-які інші вершини на шляху, окрім безпосередньо попередньої та наступної вершин є недоступною, так само як і інформація про довжину шляху і позицію даної вершини на ньому. Пакет перебудовується на кожному кроці пересилання для того, щоб зловмисник, який працює на рівні мережі, не мав змоги пов’язати між собою пакети, що належать одному й тому ж шляху.

Транзакційний шлях конструюється вихідною вершиною, якій відомі публічні ключі кожної вершини-посередника і кінцевої вершини, що дозволяє вихідній вершині створити за допомогою алгоритму ECDH спільний ключ з кожною з вершин-посередників і з кінцевою вершиною. Цей спільний ключ використовується для генерації псевдовипадкового потоку байтів для обфускації пакету та набору ключів для шифрування вмісту пакету та криптографічної перевірки цілісності даних, яка в свою чергу використовується для забезпечення цілісності пакету на кожному кроці пересилання.

Вершина на кожному переході при пересиланні пакету вздовж обраного шляху бачить лише ефемерний ключ вихідної вершини, що приховує інформацію про вихідну вершину. Ці ефемерні ключі маскуються кожною вершиною-посередником перед пересиланням

наступній вершині на шляху, що унеможливлює встановлення зв'язку між окремими "цибулинами" одного й того ж транзакційного шляху.

2.2.1. Структура пакета

Оскільки кожній вершині доступна інформація про топологію мережі, вихідна вершина конструює пакет, що повністю визначає шлях транзакції і містить всі необхідні дані для поширенням HTLC на кожному переході.

Пакет має фіксований розмір, розрахований на 20 вершин-учасників шляху, включаючи вихідну та кінцеву вершини, і має наступну структуру (комою відокремлено загальний розмір поля в байтах):

1. *Версія*, 1 – версія протоколу, значення 0x00;
2. *Публічний ключ*, 33 – ефемерний публічний ключ для першого переходу на транзакційному шляху;
3. *Дані про переходи*, $20 * 65 = 1300$ – масив елементів інформації, необхідної для поширення HTLC на кожному переході, де кожен елемент має наступну структуру:
   - *Клас*, 1 – значення, що вказує на формат, який використовується для подання даних для кожного переходу (на даний момент визначено лише значення $0x00$;
   - *Пакет переходу*, 32 – інформація, необхідна для здійснення даного переходу, формат якої задається значення *Класу*, описаним вище, і для класу $0x00$ має наступний формат:
     - *Ідентифікатор каналу*, 8 – короткий ідентифікаційний номер каналу, що належить вершині-посереднику, для якої передбачено даний пакет переходу;
     - *Сума пересилання*, 8 – кількість атомарних одиниць вартості, яку необхідно переслати даним переходом;
     - *Вихідне значення CLTV*, 4 – часове значення t для перевірки операцією Біткоїн-скрипта *Check Lock Time Verify*, відносно

якого повинен бути створений контракт HTLC з наступною вершиною на транзакційному шляху (див. пункт 1.2.2);

– *Доповнення*, 12 – поле, зарезервоване для використання у майбутніх версіях протоколу;

– *HMAC-значення*, 32 – криптографічне значення, обчислене алгоритмом *Keyed-Hash Message Authentication Code*, описаним у документі FIPS 198 Standard/RFC 2104 [10], що забезпечує цілісність та недоторканість даного пакету переходу;

4. *HMAC-значення*, 32 – значення, аналогічне значенню 3.в, але обчислене для усього пакету.

2.2.2. Підготовка криптографічних ключів

Для опису процесу конструювання пакету припустимо, що вихідна вершина S хоче маршрутизувати пакет кінцевій вершині R. Спочатку вихідна вершина обчислює шлях $I_0, I_1, ..., I_{k-1}, I_k$, де I_0 – сама вихідна вершина S, а I_n – кінцевий вершина-отримувач R. Кожна пара вершин (I_i, I_{i+1}) повинна бути пов'язана транзакційним каналом в мережі (сусідніми вершинами з точки зору графу мережі). Вихідна вершина збирає публічні ключі для вершин від I_1 до I_{k-1} і генерує випадковий 32-байтний ключ сесії. За необхідності вихідна вершина може використати асоційовані дані, тобто дані, які перевіряються на цілісність разом з пакетом, але не включаються у нього безпосередньо.

Щоб сконструювати "цибулину", вихідна вершина встановлює ефемерним приватним ключем першого переходу p_{e_1} ключ сесії p_s і обчислює з нього відповідний ефемерний публічний ключ $P_{e_1} = p_{e_1} B_{SECP256K1}$, де $B_{SECP256K1}$ – базова точка кривої *SECP256K1*. Для кожного з k переходів на обраному шляху, вихідна вершина ітеративно обчислює спільне секретне значення s_i та ефемерний ключ для наступного переходу $p_{e_{i+1}}$, використовуючи наступний алгоритм:

1. Вихідна вершина виконує алгоритм ECDH з публічним ключем наступної вершини на шляху та ефемерним приватним ключем, щоб отримати точку кривої, від якої після цього обчислюється хеш-функція *SHA256*, результат якої і є спільним секретним значенням s_i.
2. Приховуючий множник є *SHA256*-хеш-сумою послідовності байтів, отриманої конкатенацією ефемерного публічного ключа P_{e_k} та спільного секретного значення s_i.
3. Ефемерний приватний ключ для наступного переходу $p_{e_{i+1}}$ обчислюється засобом множення поточного ефемерного приватного ключа p_{e_i} на приховуючий множник.
4. Ефемерний публічний ключ для наступного переходу $P_{e_{i+1}}$ обчислюється з ефемерного приватного ключа $p_{e_{i+1}}$ засобом множення його на базову точку кривої $B_{SECP256K1}$.

2.2.3. Процес побудови пакета

Після того, як вихідна вершина обчислить всі необхідні криптографічні елементи за допомогою алгоритму, описаного вище, вона може сконструювати остаточний пакет. Побудова пакета для транзакційного шляху з r переходів потребує r 32-байтних ефемерних публічних ключів, r 32-байтних спільних секретних значень, r 32-байтних приховуючих множників та r 65-байтних структур *пакетів переходу*. Результатом побудови є 1366-байтний пакет та адреса першої вершини-посередника на обраному транзакційному шляху.

Побудова пакета виконується в порядку, зворотньому до порядку вершин на шляху, тобто дані про останній перехід заповнюються першими. Структура пакета заповнюється 1366-ма нульовими байтами. 65-байтне значення *Заповнення* генерується спеціальним алгоритмом, деталі якого опущено, з використанням спільного секретного значення.

Для кожного переходу на шляху, в зворотньому порядку до порядку вершин на шляху, вихідна вершина виконує наступний алгоритм:

1. Ключі K_ρ та K_μ генеруються з використанням спільного секретнього значення для даного переходу.
2. Поле *Інформація про переходи* зсовується вправо на 65 байтів з відкиданнями останніх 65-ти байтів, що виходять за відмітку у 1300 байтів.
3. Поля *Версія*, *Ідентифікатор каналу*, *Сума*, *Вихідне значення CLTV*, *Заповнення* та *HMAC* копіюються у наступні 65 байтів.
4. Ключ K_ρ використовується для генерації 1300 байтів з псевдо-випадкового потоку байтів, які в свою чергу використовуються для маскування поля *Дані про переходи* за допомогою операції XOR.
5. Якщо цей перехід є останнім (тобто ця ітерація алгоритму – перша), тоді залишок поля *Дані про переходи* переписується заповненням маршрутизаційної інформації.
6. Наступне значення HMAC обчислюється над конкатенацією поля *Дані про переходи* та асоційованими даними, використовуючи ключ K_μ в якості ключа, передбаченого алгоритмом HMAC.

Кінцеве значення HMAC буде використане першою вершиною-посередником на транзакційному шляху. Результатом процесу побудови пакета є послідовність байтів, що містить байт версії, ефемерний публічний ключ для першого переходу, значення HMAC для першого переходу та модифіковані за допомогою приховуючих множників *дані про переходи*.

2.2.4. Алгоритм маршрутизації

Алгоритм маршрутизації, описаний нижче, розглядає лише пакети версії 0, оскільки майбутні версії протоколу передбачають зміни в

структурі пакетів.

Для кожного отриманого пакета, поточна вершина-посередник порівнює байт версії пакету з власною версією протоколу, що підтримується і відкидає пакет, якщо його версія не підтримується. Для пакетів з версіями, що підтримуються, поточна вершина перш-за-все десеріалізує послідовність байтів у окремі поля для подальшої обробки.

Вершина-посередник повинна виконувати наступні вимоги, які дозволяють позбавити будь-яку вершину на шляху можливості повторити маршрутизацію транзакції декілька разів у спробі здійснення аналізу потоків даних:

1. Якщо ефемерний публічний ключ не є точкою на кривій *SECP256K1*:
   - *необхідно* зупинити обробку пакета;
   - *необхідно* повідомити вихідну вершину про помилку маршрутизації.
2. Якщо пакет вже пересилався раніше:
   - якщо прообраз значення R відомий:
     - *дозволено* одразу виконати HTCL-контракт, використовуючи даний прообраз,
   - інакше:
     - *необхідно* зупинити обробку пакета і повідомити вихідну вершину про помилку маршрутизації.
3. Якщо обчислене значення HMAC і значення HMAC пакета відрізняються (помилка під час перевірки цілісності):
   - *необхідно* зупинити обробку;
   - *необхідно* повідомити вихідну вершину про помилку маршрутизації.
4. Якщо значення поля *Клас* не відоме:
   - *необхідно* зупинити обробку;

– *необхідно* повідомити вихідну вершину про помилку маршрутизації.

5. *Необхідно* надсилати пакет лише безпосереднім сусідам поточної вершини в мережі.
6. Якщо поточна вершина не має сусіда, адреса якого відповідає адресі, вказаній у пакеті:
   – *необхідно* зупинити обробку;
   – *необхідно* повідомити вихідну вершину про помилку маршрутизації.

Варто звернути увагу на те, що захист від подібного аналізу методом спроб можна реалізувати, використовуючи відслідковування попередньо використаних спільних секретних значень чи значень HMAC, які можна безпечно видаляти, як тільки стає відомо, що відповідний контракт HTLC не буде прийнято (наприклад, після того, як момент часу, вказаний у полі *Вихідне значення CLTV* вже відбувся).

2.2.5. Деконструкція пакету

Після цього поточна вершина обчислює спільне секретне значення, використовуючи приватний ключ, що відповідає її публічному ключу, та ефемерний публічний ключ, що вміщений у пакеті, використовуючи алгоритм ECDH.

Далі поточна вершина використовує спільне секретне значення для того, щоб обчислити ключ K_μ, який вона в свою чергу використовує для обчислення значення HMAC для поля *Дані про переходи*. Результуюче значення HMAC порівнюється з значенням HMAC пакету, що необхідно виконувати за константний час для уникнення витоків інформації, після чого поточна вершина може обчислювати ключі K_ρ та K_γ.

Маршрутизаційна інформація демаскується, після чого поточна вершина дістає з структури інформацію про наступний перехід. Для цього

вона копіює поле *Дані про переходи* доповнює його 65-ма нульовими байтами, генерує 1365 псевдовипадкових байтів (використовуючи ключ K_ρ) і обчислює результат операції XOR від них та копії поля *Дані про переходи*. Перші 65 байтів отриманого результати і є вмістом поля *Пакет переходу* структури, що описує наступний перехід. Наступні 1300 байтів є масивом *Дані про переходи* пакету, який необхідно надіслати далі.

2.2.6. Завершення маршрутизації

Якщо значення HMAC не вказує на закінчення транзакційного шляху і наступний перехід є безпосереднім сусідом поточної вершини, виконується побудова нового пакету, яка полягає у маскуванні ефемерного ключа публічним ключем поточної вершини та спільним секретним значенням, і серіалізації структур *Дані про переходи*. Після цього отриманий пакет пересилається вказаному в маршрутизаційних даних сусіду поточної вершини.

Спеціальне значення HMAC для структури *Пакет переходу*, що складається з 32 нульових байтів, вказує на те, що поточний перехід є останнім, (тобто поточна вершина є кінцевою, отримувачем транзакції) і пакет не потрібно пересилати далі.

2.3. Транзакційна топологія

Як видно з визначеннь у підрозділі 1.3, розглянутий протокол маршрутизації транзакцій реалізує різновид цибулевої маршрутизації, використовуючи маршрутизацію від джерела в якості основного процесу. Так, вихідна вершина самостійно обирає шлях пакету і розміщує цю інформацію безпосередньо у ньому. Кожна вершина-посередник бачить призначену для неї частину маршрутної інформації і виконує пересилання пакету так, як у ньому зазначено, що за визначеннням (див. пункт 1.3.1) і

є маршрутизацією від джерела.

Ключовим недоліком цього протоколу, як і будь-якого протоколу на основі маршрутизації від джерела, є необхідність доступу до інформації про структуру мережі (наявні маршрутизатори, зв'язки між ними, тощо), оскільки вихідна вершина використовує граф мережі безпосередньо при виборі шляху транзакції. У випадку з набором протоколів TCP/IP, що лежать в основі технології Інтернет, ця інформація є важливою частиною публічної інфраструктури. У випадку мережі транзакційних каналів, такої як Lightning Network, інформація про вершини в мережі та зв'язки між ними (це стосується всіх вершин, оскільки кожна з них є маршрутизатором) має додатковий зміст: так як зв'язок між вершинами є транзакційним каналом, його можна інтерпретувати як фактичну тривалу фінансову взаємодію між двома вершинами в мережі. Таким чином, сама інформація про структуру мережі неявно містить в собі інформацію про фінансові зв'язки між учасниками мережі, яка може не мати ніякого значення сама по собі, але за наявності будь-яких асоціацій між вершинами в мережі і постачальниками певних товарів та послуг дозволяє з певною ймовірністю стверджувати про факти оплати тими чи іншими сутностями тих чи інших товарів чи послуг. Потенційні негативні і небезпечні наслідки таких асоціацій не потребують пояснення.

Для наглядної демонстрації проблеми транзакційної топології на рис. 2.2 зображено граф мережі транзакційних каналів, про одну з вершин якого (E_7) відомо, що вона підтримується агентством з продажу нерухомості, а про іншу – що це вершина, яка належить особі на ім'я Дж. Ґолт. Форма графа вказує на існування транзакційних каналів між вершинами E_3, E_4, E_5 та E_6 і вершиною-агентством нерухомості E_7, що дозволяє з достатньо високою ймовірністю стверджувати, що ці вершини здійснювали чи планують здійснити покупку нерухомості, що в свою чергу вказує на наявність достатньо великих грошових сум і

дозволяє робити подальші припущення, залежно від цілей зловмисника. Найбільшу небезпеку несуть асоціації фізичних осіб. Так, з даного графа можна зробити висновок, що у особи на ім’я Дж. Ґолт є достатня кількість грошей для покупки нерухомості. Будь-яка фінансова система, що допускає можливість витоків подібної інформації, є непридатною для повсякденного використання.

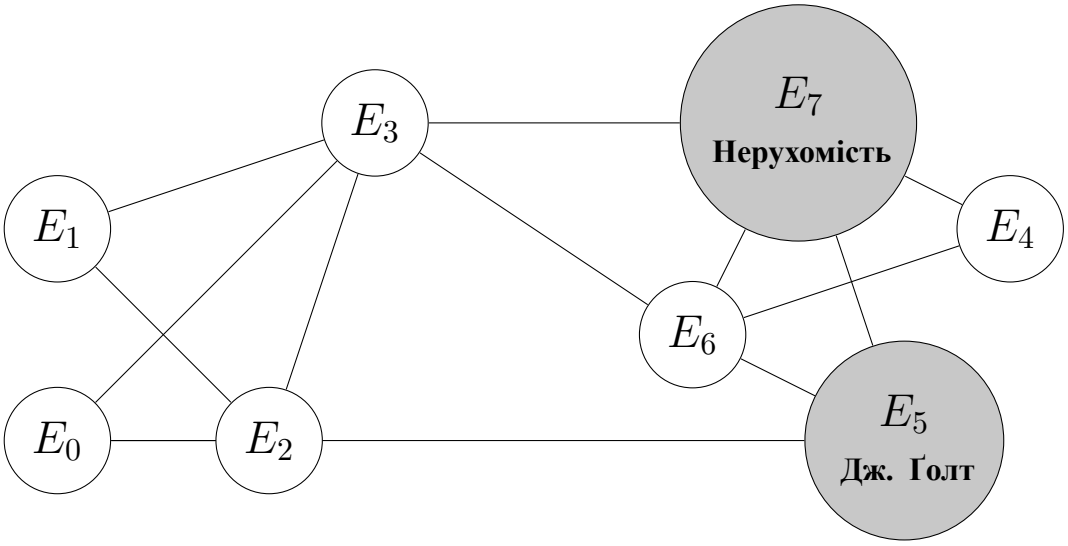


Рис. 2.2. Граф мережі з асоціаціями

Метою даного дослідження є усунення можливість витоків такої інформації за допомогою усунення інформації про наявні вершини та зв’язки з ними у публічно доступному графі мережі.

Якщо структура мережі за визначенням містить інформацію про фінансові зв’язки між її учасниками, а задача дослідження полягає у мінімізації такої інформації чи доступу до неї, очевидним рішенням є вивчення потенційної можливості організації процесу маршрутизації в умовах відсутності доступу до інформації про структуру мережі. Саме це й розглядається у наступному розділі даного дослідження.

2.4. Висновки

В даному розділі запропоновано графову модель акторів мережі криптовалютних каналів, що виділяє класи вершин мережі на основі функцій, які вони виконують, відносно конкретної транзакції. Ця модель дозволяє наглядно продемонструвати загальний процес маршрутизації, а також вводить термінологічну базу для опису цього процесу, що використовує теорію графів та класифікацію учасників мережі за функціями, які вони виконують.

На основі запропонованої графової моделі акторів мережі сформульовано поняття фінансової топології – структури графа мережі, що розглядається в контексті сфери застосування цієї мережі та за наявності зовнішньої інформації про окремих учасників мережі. Описано проблему відкритої фінансової топології мережі криптовалютних каналів Lightning, яка полягає в тому, що за наявності множини асоціацій між вершинами в мережі та сутностями, що їм відповідають, публічно доступна інформація про топологію мережі слугує джерелом витоків даних про фінансові зв’язки між цими сутностями, що порушує приватність їхньої фінансової інформації.

3. ЗАПРОПОНОВАНИЙ МЕТОД МАРШРУТИЗАЦІЇ ТРАНЗАКЦІЙ

3.1. Мережа криптовалютних каналів з прихованою топологією

Оскільки відкриття транзакційного каналу є взаємодією між двома окремими сутностями і фізично відображається лише транзакцією в основній мережі Bitcoin, можливо побудувати мережу так званих приватних каналів. Така мережа нічим не відрізняється від звичайної мережі Lightning, але інформація про канали в ній не публікується.

3.1.1. Приватні транзакційні канали

Розглянемо мережу приватних транзакційних каналів зображену на рис. 3.1. Даний граф демонструє мережу з точки зору вершини E_5: оскільки ця вершина знає про відкриті нею канали з вершинами E_2, E_4 та E_6, вона знає лише про існування цих трьох вершин. Решта мережі може складатись з довільної кількості вершин та зв'язків між ними, але на відміну від мережі Lightning, описаної у розділі 2, ця інформація є прихованою від вершини E_5, і її власна модель мережі, яку називають "уявленням" (англ. *View*), складається лише зі згаданих 3-х вершин.

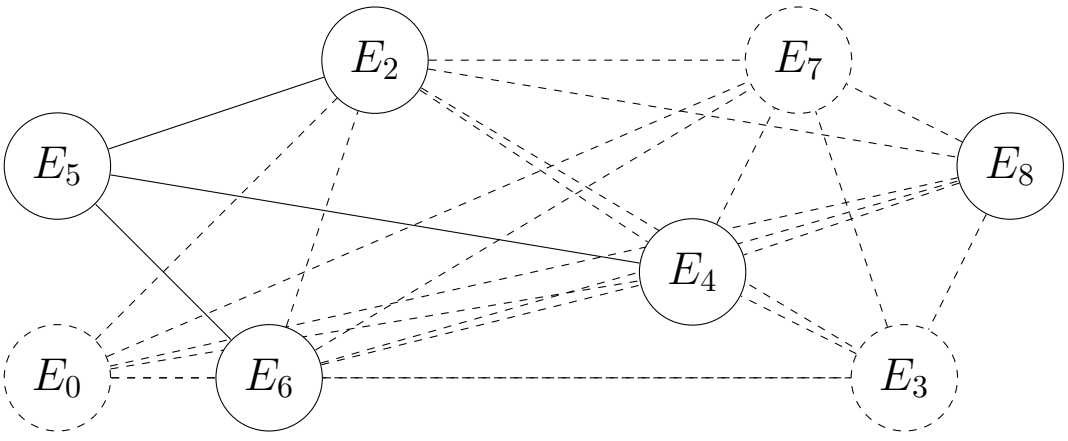


Рис. 3.1. Мережа приватних транзакційних каналів

Припустимо, що вершина E_5 встановлює позамережевий зв'язок з вершиною E_8 з метою здійснити транзакцію (під позамережевим зв'язком

мається на увазі обмін повідомленнями за допомогою будь-якого іншого каналу). В цей момент в уявленні вершини E_5 з'являється нова вершина E_8, яка при цьому не має жодного зв'язку. Основна гіпотеза даного дослідження у цьому конкретного випадку полягає в тому, що за умови зв'язності мережі інформації про множину вершин $\{E_2, E_4, E_6\} \cup \{E_8\}$ достатньо для здійснення маршрутизації транзакції від E_5 до E_8.

Мережа приватних транзакційних каналів має значну перевагу над мережею публічних каналів Lightning з точки зору її застосування: вона усуває з мережі метадані про фінансові зв'язки між сутностями.

3.1.2. Приховування транзакційної топології

Як вже було зазначено раніше, кожен зв'язок в мережі означає фінансове відношення між двома сутностями в мережі. Практично будь-яка фінансова взаємодія виходить за межі цифрового простору: переважна більшість товарів та послуг постачаються чи надаються поза межами мережі Інтернету, тому фінансові взаємодії в тій чи іншій мірі пов'язані з фізичним світом. Як наслідок, будь-яка інформація про фінансові зв'язки є "токсичною", тобто такою, що може бути використана для отримання різноманітної приватної інформації про сторони цих зв'язків.

Оскільки мережа приватних транзакційних каналів усуває інформацію про фінансові зв'язки між сутностями, вона є більш безпечною для використання. Факт існування вершини в мережі не може бути використаний для аналізу її фінансових потоків, оскільки з точки зору будь-якої іншої вершини вона не має ніяких зв'язків. Отже, мережа приватних транзакційних каналів, або мережа транзакційних каналів з прихованою топологією, вирішує проблему фінансової топології, сформульвану в розділі 2. Втім, для того, щоб така мережа могла використовуватись, необхідно продемонструвати можливість роботи в ній

транзакційного протоколу, визначеного в розділі 2.

В наступному підрозділі (3.2) описується процес підготовки шляху транзакції у мережі приватних транзакційних каналів, який відповідає процесу побудови пакета в оригінальному протоколі, а в підрозділі 3.3 розглядається повний алгоритм маршрутизації в такій мережі.

3.2. Підготовка шляху

Для забезпечення локальних характеристик безпеки даних транзакції протокол, що пропонується, використовує цибулеву маршрутизацію аналогічно до оригінального протоколу мережі Lightning, описаного в [11].

Для схематичного зображення компонентів протоколу тут і надалі розглядається мережа з 8-и вершин, зображена на рис. 3.2.

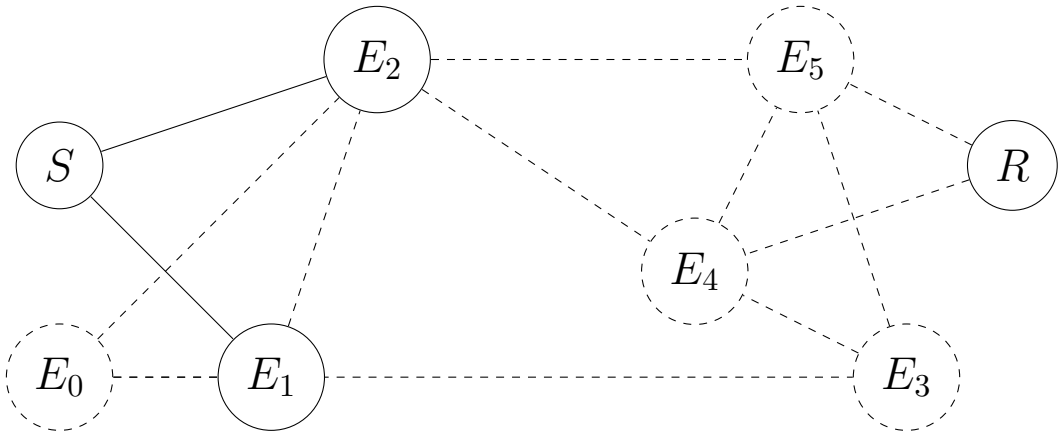


Рис. 3.2. Приклад мережі приватних каналів

Як було зазначено у пункті 2.2.3, для побудови пакета-цибулини початковій вершині S необхідно зібрати впорядковану множину $\{P_{I_1}, P_{I_2}, ..., P_{I_n}\}$ публічних ключів вершин-посередників для шифрування даних. Згідно з специфікацією оригінального протоколу, вихідна вершина отримує інформацію про шлях до кінцевої вершини з свого уявлення про мережу – структури даних, що містить інформацію про всі публічні вершини, їхні адреси та активні публічні канали – зв'язки. В умовах

мережі приватних транзакційних каналів вихідна вершина за визначенням не має уявлення про мережу, єдина доступна їй інформація – факти існування її сусідніх вершин, тобто вершин E_1 та E_2, з якими вона має безпосередні транзакційні канали, наявність зв'язків з сусідніми вершинами та очевидний факт існування кінцевої вершини R. Через це процес побудови шляху є більш складним і, на відміну від простого пошуку шляху в графі для оригінального протоколу, полягає у взаємодії з учасниками мережі.

3.2.1. Пошук шляху

Перед здійсненням пошуку шляху початкова та кінцева вершини виконують підготовку до маршрутизації для узгодження умов транзакції та встановлення параметрів пошуку.

Процес підготовки відбувається поза межами мережі транзакційних каналів з використанням каналу зв'язку, що вважається криптографічно безпечним, що зображено на рис. 3.3.

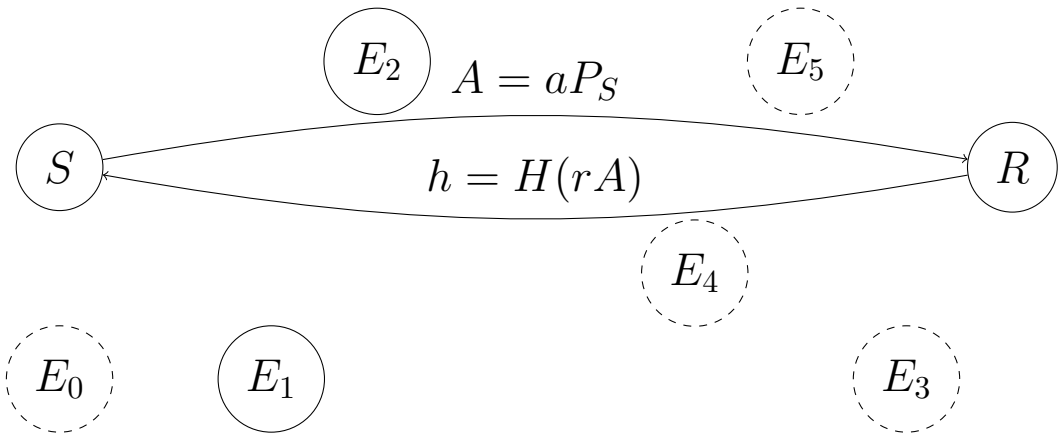


Рис. 3.3. Підготовка до пошуку шляху

Початкова та кінцеві вершини виконують алгоритм ECDH для встановлення спільного секретного значення: початкова вершина генерує випадкове значення a достатнього розміру для забезпечення

криптографічної безпеки від вгадування (згідно з рекомендаціями для цього достатньо 256 біт ентропії), обчислює значення $A = aP_S$, де P_S – її публічний ключ для даної транзакції і передає значення A кінцевій вершині, яка в свою чергу генерує випадкове значення r, і обчислює значення $h = H(rA)$, де H – криптографічно безпечна хеш-функція (для кращої інтеграції в набір протоколів Bitcoin рекомендується використання функції *SHA256*, яка використовується у багатьох компонентах даної технології). Значення h передається початковій вершині S для виконання алгоритму побудови шляху.

Вихідна вершина конструює пошуковий пакет (h, P_S, T), де P_S – публічний ключ вихідної вершини, і надсилає його кожному з своїх безпосередніх сусідів. Цей крок ініціює процес конструювання маршруту, а T – додаткова інформація для обмеження тривалості поширення пакета в мережі.

Кожна вершина підтримує таблицю з пошуковими ключами r_i, що відповідають платежам, які вона очікує, та обробляє вхідні пошукові пакети наступним чином:

1. Якщо отримане з пакету значення h не відповідає жодному певному значенню r_i з таблиці пошукових ключів даної вершини, вона створює новий пакет

$$Req' = (h, P_S, T'),$$

   який надсилає всім своїм безпосереднім сусідам окрім того, хто передав даний пакет. При цьому поточна вершина зберігає в локальному стані інформацію про те, який з її безпосередніх сусідів передав їй даний пакет. Слід звернути увагу на те, що значення T, яке використовується для забезпечення завершення поширення пакета, не може базуватись на стандартній техніці “час життя” (англ. *TTL*, *Time to live*), яка полягає в тому, що

значення декрементується кожним маршрутизатором на шляху, і пакет не передається далі, якщо його час життя дорівнює 0, таким чином поширюючись в мережі лише на фіксований радіус від вихідної вершини. Використання цієї техніки дозволило б вершинам в мережі визначати відстань від них до вершини, що є автором пакета. Однією з альтернатив може бути підхід, що базується на часових обмеженнях, на зразок заголовка `Expires:`, передбаченого протоколом HTTP [12].

2. Якщо отримане значення h відповідає певному значенню r_i з таблиці пошукових ключів даної вершини, процес пошуку завершується успіхом, і дана вершина є шуканою кінцевою вершиною R. З цього моменту починається етап повернення пошукового результату. Для цього кінцева вершина генерує $L - 1$ ефемерних пар ключів (p_{e_i}, P_{e_i}), де L – параметр мережі, що встановлює максимальну довжину шляху (для мережі Lightning $L = 20$), і повертає вершині, яка передала їй даний пошуковий пакет, пошуковий результат, що містить пошуковий ідентифікатор h та L точок еліптичної кривої:

$$Resp = (h, R, P_R, P_{e_1}, ..., P_{e_{L-1}}),$$

де

P_R – її публічний ключ для даної транзакції,

$R = rP_S$ – доказ коректності пошуку,

а ключі P_R, P_{e_1} ..., $P_{e_{L-1}}$ формують ефемерну множину ключів для маршрутизації транзакцій. Використання $L - 1$ додаткових ключів забезпечує фіксований розмір пакета-результату пошуку, через що жодна з вершин, що беруть участь в поверненні результату, не може визначити свою позицію у побудованому шхяху.

Процес пошуку шляху до кінцевої вершини зображено на рис. 3.4. Цифрами позначено крок пошуку, жирним виділено маршрут, який досягнув кінцевої вершини найвшидше.

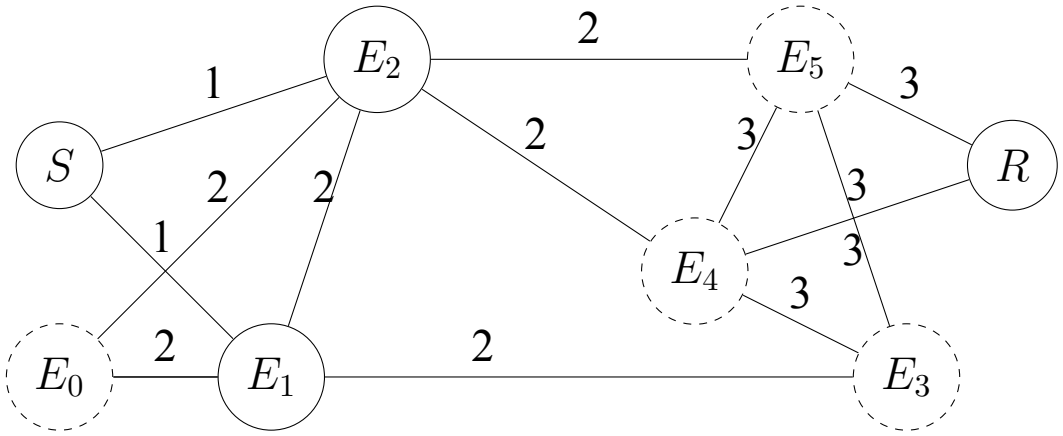


Рис. 3.4. Поширення пошукового пакета в мережі

3.2.2. Побудова маршруту

В процесі повернення пошукових результатів пакети-відповіді отримують лише вершини, які будуть виконувати роль посередників, а отже є членами утвореного транзакційного шляху, і повинні повідомити вихідній вершині ключі, необхідні для продовження виконання протоколу. Кожна вершина обробляє вхідні пакети наступним чином:

1. Якщо отримане з пакету значення h не відповідає пошуку, який був ініційований даною вершиною, дана вершина набуває ролі вершини-посередника I для даної транзакції і повинна додати свій ключ в множину ключів пакету-результату пошуку. Для цього вона розпаковує результат і отримує послідовність ключів P_1, P_2, ..., P_L, після чого формує новий пакет-відповідь

$$Resp' = (h, R, P_I, P_1, ..., P_{L-1}),$$

де P_I – її ключ для даної транзакції, таким чином здійснюючи зсув послідовності ключів вправо на одну позицію і відкидаючи

ключ P_L . Слід звернути увагу на те, що дана вершина не знає своєї позиції на шляху пакета, і тому вона не може відрізнити справжніх ключів вершин на даному шляху від ефемерних ключів, згенерованих кінцевою вершиною. Після цього поточна вершина здійснює пошук за значенням h в своїй таблиці пошуку і отримує адресу одного з своїх безпосередніх сусідів, який передав відповідний пошуковий пакет, і повертає йому новий пакет-відповідь. Інформація про відповідність пошукового пакета, вершини-сусіда, який його передав, та вершини-сусіда, який передав відповідь на нього, зберігається в локальній маршрутизаційній таблиці у вигляді трійки (h, N_p, N_n), де N_p – ідентифікатор каналу з сусідньою вершиною, яка передав пошуковий пакет, а N_n – ідентифікатор каналу з сусідньою вершиною, яка передала пакет-відповідь, таким чином формуючи множину ланок маршруту для даної транзакції. В оригінальному методі маршрутизації ідентифікатор наступного каналу входить в пакет переходу (див. пункт 2.2.1), але оскільки вершини в мережі з прихованою топологією знають лише про існування своїх власних каналів, запропонований метод маршрутизації розподіляю цю інформацію між відповідними вершинами-маршрутизаторами.

2. Якщо отримане з пакету значення h відповідає пошуку, який був ініційований даною вершиною, вона здійснює перевірку коректності знайденого шляху, обчислюючи значення $K_i = aR$, і перевіряючи рівність

$$H(K) = H(aR) = H(arP_s) = H(rA) = h$$

Якщо ця перевірка спрацьовує, процес побудови шляху був завершений успішно, і в отриманому пакетів міститься послідовність ключів, необхідних для виконання основного

протоколу маршрутизації, описаного в пункті 2.2.4.

Варто звернути увагу на те, що описаний вище процес окрім збору ключів виконує побудову маршрутизаційних таблиць для подальшого використання в процесі пересилання транзакцій. Такі записи в таблиці є одноразовими, оскільки стосуються не окремих сутностей в мережі, а одноразових ключів, що є дійсними лише для даної конкретної транзакції. Хоча така маршрутизація може вважатись табличною, вона суттєво відрізняється від табличної маршрутизації, описаної у пункті 1.3.2, таблиці якої містять записи що стосуються конкретних сутностей (маршрутизаторів та користувачів) і є багаторазовими, оскільки потребують перебудови лише при змінах топології мережі.

Процес повернення пакету-відповіді та конструювання маршруту зображено на рис. 3.5.

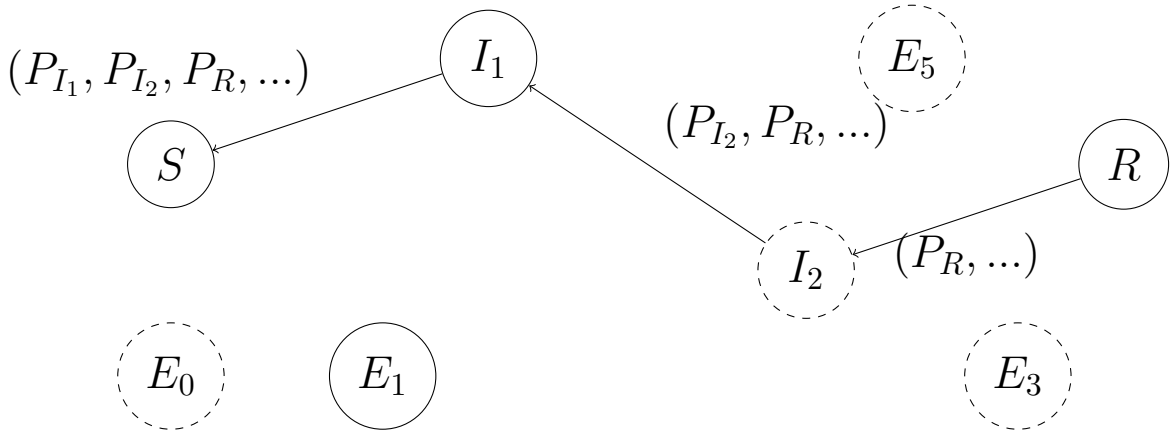


Рис. 3.5. Конструювання маршруту

3.3. Процес маршрутизації

У даному підрозділі формулюється процес сліпої маршрутизації транзакцій, який базується на оригінальному протоколі маршрутизації транзакцій від джерела, описаному в пункті 2.2.4, але містить модифікації, необхідні для роботи в мережі з прихованою топологією. Детальні

пояснення величин, що використовуються, вміщені в попередньому підрозділі. Для спрощення опису алгоритмів припускається наступне:

1. Всі асиметричні криптографічні операції здійснюються над точками еліптичної кривої *SECP256K1* [13].
2. Всі вершини мають одноразові пари ключів (p, P).
3. Між вершинами мережі існують безпечні канали зв'язку (безпосередня TLS-комунікація, тощо).

3.3.1. Алгоритм Request

Алгоритм *Request* – взаємодія між початковою та кінцевою вершинами для встановлення параметрів транзакції:

1. Вершина S:
   - генерує випадковий скаляр a;
   - обчислює значення $A = aP_S$;
   - передає значення A вершині R.
2. Вершина R:
   - генерує випадковий скаляр r;
   - обчислює значення $h = \mathrm{SHA256}(rA) = \mathrm{SHA256}(raP_S)$;
   - обчислює значення $H = \mathrm{SHA256}(r)$;
   - передає значення h та H вершині S.

Варто звернути увагу на те, що одне й те ж випадкове значення r використовується і для доказу існування шляху, і як образ хеш-функції для побудови контрактів HTLC. Це зменшує кількість криптографічного матеріалу, який необхідно зберігати.

3.3.2. Алгоритм Send

Алгоритм *Send* – взаємодія початкової вершини з усіма вершинами мережі для пошуку шляху, що завершується взаємодією початкової вершини з першою вершиною на шляху для початку надсилання

транзакції:

1. Вершина S:
   - формує пошуковий пакет $Req = (h, P_S)$;
   - надсилає пошуковий пакет Req всім своїм сусідам N_i.
2. Кожна вершина E_i в мережі:
   - отримує пакет $Req = (h, P_S)$ від сусіда N_p;
   - якщо значення h повторюється:
     - відкидає пакет Req;
   - інакше:
     - створює маршрутну вказівку (h, N_p);
     - надсилає пакет Req всім сусідам $\mathcal{N}_{\mathcal{E}_\rangle} \setminus \{N_p\}$.
3. Вершина R:
   - отримує пакет $Req = (h, P_S)$ від сусіда N_p;
   - знаходить значення r, відповідне значенню h;
   - генерує L одноразових ключів P_i;
   - обчислює значення $R = rP_S$;
   - формує відповідь $Resp = (h, R, P_R, P_1, ..., P_{L-1})$;
   - надсилає відповідь $Resp$ сусіду N_p;
4. Кожна вершина E_i в мережі:
   - отримує пакет $Resp = (h, R, P_1, ..., P_L)$ від сусіда N_n;
   - конструює новий пакет $Resp' = (h, R, P_{E_i}, P_1, ..., P_{L-1})$;
   - знаходить маршрутну вказівку (h, N_p);
   - створює нову маршрутну вказівку (h, N_n);
   - надсилає пакет $Resp'$ сусіду N_p.
5. Вершина S:
   - отримує пакет $Resp = (h, R, P_1, ..., P_L)$ від сусіда N_n;
   - знаходить значення a, відповідне значенню h;
   - якщо $\mathrm{SHA256}(aR) = \mathrm{SHA256}(rA) = h$:
     - знаходить значення H, відповідне значенню h;

- встановлює параметром HTLC значення H;
- конструює транзакційний пакет згідно з пунктом 2.2.3;
- надсилає транзакційний пакет сусіду N_n;

– інакше:

- сигналізує помилку пошуку шляху.

Як було зазначено, під час виконання даного алгоритму в мережі використовують два типи пакетів: пошуковий пакет та пакет шляху.

Пошуковий пакет має наступну структуру (комою виділено розмір поля в байтах):

1. *Версія*, 1 – версія протоколу, значення 0x01.
2. *Тип пакета*, 1 – тип пакета, необхідний для визначення етапу маршрутизації, значення 0x00 для типу *Пошуковий пакет*.
3. *Пошуковий ідентифікатор*, 32 – значення h, що ідентифікує запит на здійснення транзакції і використовується для перевірки наявності шляху.
4. *Публічний ключ*, 33 – ефемерний публічний ключ вихідної вершини для перевірки наявності шляху.

Пакет шляху має наступну структуру:

1. *Версія*, 1 – версія протоколу, значення 0x01.
2. *Тип пакета*, 1 – тип пакета, необхідний для визначення етапу маршрутизації, значення 0x01 для типу *Пакет шляху*.
3. *Пошуковий ідентифікатор*, 32 – значення h, що ідентифікує запит на здійснення транзакції і використовується для побудови шляху та створення маршрутних вказівок.
4. *Доказ існування шляху*, 32 – значення, яке підтверджує, що пошук шляху досягнув кінцевої вершини.
5. *Ключі вершин-посередників*, $20 * 33 = 660$ – масив ефемерних ключів для здійснення цибулевої маршрутизації.

3.3.3. Алгоритм Forward

Алгоритм *Forward* – послідовна взаємодія між вершинами на встановленому транзакційному шляху для створення нових контрактів HTLC, що в разі успішної маршрутизації завершується “стягуванням” коштів в зворотньому напрямку:

1. Кожна вершина I_i на транзакційному шляху:
   - отримує транзакційний пакет з параметром HTLC h;
   - деконструює пакет згідно з пунктом 2.2.5;
   - отримує з пакета параметр HTLC H
   - отримує з пакета суму пересилання для даного переходу;
   - знаходить маршрутну вказівку (h, N_n);
   - створює контракт HTLC відносно параметра H з сусідом N_n;
   - передає пакет сусіду N_n згідно з пунктом 2.2.5.
2. Вершина R:
   - отримує пакет з параметром HTLC H від сусіда N_p;
   - знаходить значення r, відповідне значенню h;
   - надає значення r сусіду N_p, виконуючи контракт HTLC.
3. Кожна вершина I_i на транзакційному шляху:
   - отримує значення r від сусіда N_n;
   - оновлює на основі r контракт HTLC з сусідом N_n;
   - надає значення r сусіду N_p, виконуючи контракт HTLC.

Структура транзакційного пакета дещо відрізняється від структури пакета, що використовується в оригінальному методі маршрутизації, оскільки в даному випадку поля на зразок *Ідентифікатор каналу* не мають змісту, так як канали є прихованими, і тому ідентифікатори каналів зберігаються у вигляді тимчасових маршрутних вказівок у локальних таблицях вершин-посередників.

Транзакційний пакет має наступну структуру:

1. *Версія*, 1 – версія протоколу, значення 0x01;
2. *Тип пакета*, 1 – тип пакета, необхідний для визначення етапу маршрутизації, значення 0x02 для типу *Транзакційний пакет*.
3. *Ідентифікатор транзакції*, 32 – значення h, що використовується при відновленні шляху як ключ у таблиці маршрутних вказівок для пошуку ідентифікатора каналу з наступною вершиною на транзакційному шляху.
4. *Дані про переходи*, $20 * 57 = 1300$ – масив елементів інформації, необхідної для поширення HTLC на кожному переході, де кожен елемент має наступну структуру:
   - *Клас*, 1 – значення, що вказує на формат, який використовується для подання даних для кожного переходу (на даний момент визначено лише значення $0x00$;
   - *Пакет переходу*, 32 – інформація, необхідна для здійснення даного переходу, формат якої задається значення *Класу*, і для даного методу маршрутизації відрізняється від пакету переходу оригінального методу тим, що не містить ідентифікатора каналу:
     - *Сума пересилання*, 8 – кількість атомарних одиниць вартості, яку необхідно переслати даним переходом;
     - *Вихідне значення CLTV*, 4 – див. пункт 2.2.1;
     - *Доповнення*, 12 – поле, зарезервоване для використання у майбутніх версіях протоколу;
   - *HMAC-значення*, 32 – див. пункт 2.2.1;
5. *HMAC-значення*, 32 – див. пункт 2.2.1.

Можливим варіантом спрощення запропонованого методу є заміна доказу існування шляху на основі обчислення хеш-функції від спільного секретного значення, встановленого за допомогою алгоритму ECDH, на використання додавання точок кривої, що дозволить використовувати образ H в якості ідентифікатора транзакції замість значення h.

3.4. Висновки

У даному розділі розглядається модель мережі транзакційних каналів, що не передбачає доступу до інформації про структуру мережі. Така мережа складається з вершин, які пов’язані між собою прихованими каналами. Таким чином, хоча дана мережа є повнозв’язною, про існування того чи іншого зв’язку знають лише вершини на кінцях відповідного йому ребра. Таку мережу приватних транзакційних каналів можна назвати *мережею з прихованою топологією*.

Використовуючи модель мережі транзакційних каналів з прихованою топологією, було запропоновано метод маршрутизації, що не потребує доступу до інформації про структуру мережі при побудові маршруту, натомість замінюючи маршрутизацію від джерела використанням тимчасових маршрутизаційних таблиць. Такий підхід можна назвати “сліпою маршрутизацією”, оскільки кожна вершина, що бере участь в побудові шляху, не отримує ніякої інформації про зв’язки з вихідною та кінцевою вершинами, окрім того, який з її безпосередніх сусідів “ближчий” до кінцевої вершини.

4. АНАЛІЗ ТА РЕАЛІЗАЦІЯ ЗАПРОПОНОВАНОГО МЕТОДУ

4.1. Властивості та недоліки запропонованого методу

Основною ціллю даного дослідження була розробка методу маршрутизації, що дозволяє здійснювати транзакції в мережі з прихованою топологією. Втім, за визначенням, процес маршрутизації використовує топологію мережі, оскільки пакети передаються між учасниками мережі вздовж ребер графа.

Фундаментальна залежність процесу маршрутизації від топології мережі означає, що форма графа мережі може бути відновлена з метаданих, що генеруються в процесі маршрутизації пакета. В пункті 4.1.1 розглядається формальна задача відновлення структури графа мережі та демонструється неможливість її розв'язку в межах описаної моделі.

Окрім того, приховання інформації про топологію мережі від кожної окремої вершини насправді означає, що ця інформація є розподіленою між множиною всіх вершин, а отже будь-які обчислення, що використовують частини інформацію про топологію (в даному випадку – прийняття рішень під час маршрутизації) делегуються до вершин, що мають доступ до цих частин інформації. Це забезпечує захист від агрегації даних, але збільшує кількість обчислювальних ресурсів, необхідних для виконання маршрутизації і сповільнює сам процес. Аналіз цих недоліків подано в пункті 4.1.2.

4.1.1. Приховування топології мережі

Розглянемо функцію $Q_{\mathcal{G}} : \mathcal{N} \times \mathcal{N} \mapsto \{0, 1\}$, яка повертає булеве значення про існування вершини інформацію, яку вихідна вершина S отримує в результаті пошуку шляху:

$$Q_{\mathcal{G}}(n_i, n_j) = \begin{cases} 1, & \text{якщо між вершинами } n_i \text{ та } n_j \text{ існує шлях у графі } \mathcal{G}, \\ 0 & \text{в усіх інших випадках.} \end{cases}$$

Мережа приватних транзакційних каналів забезпечує надійне приховування даних про фінансові зв'язки між сутностями-учасниками мережі, якщо справджується наступне твердження:

> Неможливо відновити структуру графа $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, маючи множину вершин $\mathcal{N}' \subset \mathcal{N}$ і підмножину образу функції $\{Q_{\mathcal{G}}(n_i, n_j); n_i, n_j \in \mathcal{G}'\}$.

Оскільки розглядається лише підмножина $\mathcal{N}'$ вершин графа $\mathcal{G}$, решта вершин графа $\mathcal{N} \setminus \mathcal{N}'$ є невідомою, тому для будь-якого відомого значення функції $Q_{\mathcal{G}}(n_i, n_j) = 1$ неможливо визначити довжину шляху l зв'язку, на який вказує значення функції, які і значення нерівності $l > 1$, (тобто встановити, чи зв'язок є безпосереднім, що вкаже на існування ребра (n_i, n_j)).

Для того, щоб ця властивість зберігалась, необхідно виконання умови $|\mathcal{G}'| \ll |\mathcal{G}|$, тобто відомий підграф $\mathcal{G}'$ має бути значно меншим за весь граф мережі $\mathcal{G}$. Отже, основною вимогою для того, щоб мережа приватних транзакційних каналів мала приховану топологію, є достатньо велика кількість учасників. Така вимога є аналогічною до вимог безпеки різноманітних технологій децентралізованої взаємодії, зокрема мережі цибулевої маршрутизації Tor [14], протоколу децентралізованого обміну файлами BitTorrent [15] та самої криптовалютної мережі Bitcoin.

4.1.2. Недоліки запропонованого методу

Як вже було зазначено раніше, основною відмінністю даного протоколу маршрутизації транзакцій від протоколу, що використовується в існуючій мережі Lightning, є те, що процес пошуку шляху, який є частиною алгоритму ініціалізації, є інтерактивним і полягає у розсиланні пошукового запиту по всій мережі та очікуванні відповіді від кінцевої вершини, процесі передачі якої відбувається збір необхідних криптографічних матеріалів

(ключів) для здійснення транзакції. Натомість, в оригінальному протоколі пошук шляху зводиться до виконання будь-якого класичного алгоритму пошуку шляху в структурі даних, що відображає уявлення даної вершини про граф мережі.

Оскільки етапи пошуку та побудови шляху запропонованого методу є інтерактивними, вони створюють навантаження на мережу, так як потребують додаткової обробки відповідних пакетів багатьма вершинами. Як було зазначено раніше, використання традиційних засобів обмеження тривалості поширення пакета на зразок лічильника часу життя порушує приховування топології мережі, оскільки у випадку використання простого лічильника інформація про відстань до початкової вершини отримується безпосередньо з лічильника, а у випадку використання часових відміток для цього можна застосувати аналіз часових змін. Таким чином, найпростішим рішенням є повне поширення пошукового пакета по мережі, що призводить до значного підвищення трафіку в такій мережі порівняно з мережею з відкритою топологією та маршрутизацією “від джерела”.

Оцінюючи швидкодію запропонованого методу, можна розглядати довжину шляху як параметр оцінки, а передачу та обробку будь-якого пакета між двома сусідніми вершинами – як абстрактну операцію одиничної тривалості. На етапах пошуку шляху до кінцевої вершини та повернення відповіді до початкової вершнини пакети двічі проходять той же шлях, який буде використаний на етапі надсилання транзакції, тому можна зробити висновок, що запропонований метод в середньому працює втричі повільніше, ніж метод маршрутизації “від джерела”.

4.2. Програмна реалізація запропонованого методу

Прототипна реалізація запропонованого протоколу була здійснена на основі програмного забезпечення LND [16] – найбільшої за кількістю

учасників та різноманіттям компонентів реалізації набору протоколів Lightning, робота над якою розпочалась з моменту появи оригінальної публікації [2]. Дане програмне забезпечення використовує програмне середовище Golang як існовний інструмент розробки і має високий рівень модуляризації, що дозволило легко ідентифікувати компоненти, що підлягають модифікації для забезпечення правильної роботи описаного протоколу.

4.2.1. Приватні транзакційні канали

Як вже було зазначено раніше, приватним транзакційним каналом у мережі Lightning називають такий канал, про який відомо лише його безпосереднім учасникам. Такий канал можна створити, використовуючи існуюче програмне забезпечення LND, оскільки для цього достатньо створити звичайний транзакційний канал і не реєструювати його у мережі. Для цього в прикладному програмному інтерфейсі, що надається LND, передбачено прапорець під назвою *private*, який дозволяє зручно створювати такі канали з метою обмінювати транзакціями, використовуючи лише протокол транзакційного каналу, і не виконувати транзакційного протоколу мережі, таким чином не беручи участі в маршрутизації транзакцій інших учасників.

Використовуючи цей компонент існуючої реалізації набору протоколів, можна розгорнути приховану мережу, що складається виключно з приватних транзакційних каналів і використовує запропонований протокол для маршрутизації транзакцій через ці канали, при цьому повністю приховуючи свою позицію та зв’язки в мережі.

4.2.2. Підмережа приватних каналів

Як вже було зазначено, однією з основних вимог приватності даних про фінансові зв’язки є розмір мережі приватних каналів.

Оскільки існуюче програмне забезпечення LND дозволяє створення приватних каналів для безпосереднього використання, для забезпечення наявності великої кількості вершин у мережі з прихованою топологією з моменту її запуску можна реалізувати цю мережу як частину основної мережі Lightning. Для досягнення цього основною необхідною зміною в існуючому програмному забезпеченні є виділення компонентів, що беруть участь у маршрутизації транзакцій, у окремі об’єкти і розробка універсального інтерфейсу, що дозволить динамічно замінювати такі об’єкти під час роботи мережі, залежно від вхідного пакету.

Оскільки запропоновані зміни в транзакційному протоколі стосуються лише процесу маршрутизації транзакцій, програмне забезпечення, що реалізує протоколи мережі Lightning, може підтримувати множину об’єктів-маршрутизаторів, кожен з яких реалізує той чи інший протокол маршрутизації, і обирати їх для обробки конкретного пакета, залежно від версії, вказаної безпосередньо у пакеті (див. [11]). Оскільки на даний момент передбачено лише одни режим маршрутизації, який обробляє пакети, позначені версією 0, і відкидає пакети з будь-якою інші версією (див. пункт 2.2.4), сутність, що реалізує протокол сліпої маршрутизації транзакцій, може обробляти лише пакети, позначені версією 1, що дозволить уникнути будь-яких конфліктів між протоколами. Всі майбутні протоколи маршрутизації можуть використовувати очевидний аналогічний підхід, а саме визначення способу маршрутизації, залежно від версії пакета.

Таким чином кожна вершина, залежно від заданої оператором конфігураці, матиме кілька режими роботи, в першому з яких вона бере участь у стандартній маршрутизації вздовж публічних каналів, використовуючи стандартний об’єкт-маршрутизатор, у другому – маршрутизує пакети виключно вздовж приватних каналів, використовуючи об’єкт-маршрутизатор для підмережі з прихованою

топологією, а у всіх інших режимах, за тим же принципом, здійснює маршрутизацію за допомогою відповідного маршрутизатора, залежно від версії пакета.

Описана вище система з гнучким механізмом маршрутизації, що визначається версією чи типом пакета, дозволить експериментувати з різноманітними модифікаціями транзакційного протоколу в межах однієї існуючої мережі, при цьому участь в різних типах протоколу може визначатись оператором відповідної вершини за допомогою конфігурації на вибір.

4.2.3. Автоматичний вибір маршрутизатора

Як вже згадувалось вище, для реалізації запропонованого методу як розширення існуючої мережі транзакційних каналів було взято за основу програмне забезпечення LND. Було виділено інтерфейс під назвою `Router`, що надає наступний перелік методів:

1. `LookupPath` – метод, що приймає параметром значення `id` типу `string`, і повертає послідовність ключів вершин-посередників, необхідних для надсилання транзакції.
2. `SendTransaction` – метод, що приймає список ключів вершин-посередників і транзакційні дані (кількість коштів, тощо), формує транзакційний пакет і надсилає його першій вершині-посереднику, яка, за визначенням, є одним з безпосередніх сусідів даної вершини.
3. `HandlePacket` – метод, що приймає отриманий пакет та ідентифікатор безпосереднього сусіда, який його передав, і обробляє його відповідно до його типу.

Реалізація інтерфейсу `Router` для оригінального методу маршрутизації – `BasicRouter`:

1. `LookupPath` – приймає параметром публічний ключ кінцевої

вершини і здійснює пошук шляху в локальній структурі даних, що представляє граф мережі, використовуючи будь-який алгоритм пошуку шляху в графі.

2. `SendTransaction` – приймає список ключів вершин-посередників і транзакційні дані, формує транзакційний пакет відповідно до опису в пункті 2.2.1 і надсилає його першій вершині в списку посередників.
3. `HandlePacket` – якщо версія пакету не відповідає визначеній для даного методу маршрутизації версії (0), надсилає сигнал про помилку, інакше знімає верхній шар шифрування “цибулини” пакета, отримує інформацію про наступну вершину та деталі транзакції, необхідні для формування нового контракту HTLC, здійснює поширення HTLC та пересилає вкладений пакет далі.

Реалізація інтерфейсу `Router` для запропонованого методу маршрутизації – `HiddenRouter`:

1. `LookupPath` – приймає параметром значення хеш-функції `h`, отримане від кінцевої вершини, у кодуванні *Base64*), ініціює перший етап пошуку шляху запропонованого методу і очікує завершення другого етапу побудови шляху (отримання пакета-відповіді), після чого здійснює перевірку отриманої послідовності ключів і повертає її, якщо перевірка успішна.
2. `SendTransaction` – успадковано з реалізації `HiddenRouter`.
3. `HandlePacket` – якщо версія пакета не відповідає визначеній для даного методу маршрутизації версії, надсилає сигнал про помилку, інакше, залежно від типу пакета:
   – *Запит* – якщо запит містить пошуковий ідентифікатор, що зустрічається *не* вперше, відкидає пакет, як недійсний, інакше якщо пошуковий ідентифікатор стосується даної вершини, надсилає відповідь сусіду, що передав цей пакет, інакше

надсилає пакет всім своїм безпосереднім сусідам, окрім того, хто передав пакет, тимчасово зберігаючи ідентифікатор пакета та сусіда, що його передав.

– *Відповідь* – якщо пакет містить пошуковий ідентифікатор, що надсилався даною вершиною, надіслати сигнал про завершення другого етапу побудови шляху (що спричинить завершення відповідного методу `LookupPath`), інакше додає власний ключ в пакет-відповідь, знаходить в тимчасовій таблиці маршрутизації інформацію про вершину, яка передала пошуковий пакет з таким ідентифікатором і передає їй новий пакет-відповідь, зберігаючи інформацію про вершину, що передала пакет-відповідь, у таблиці маршрутизації разом з вершиною, що передала пакет-запит;
– *Транзакція* – якщо ідентифікатор транзакції присутній у локальній таблиці маршрутизаці, і вказує на вершину, що передала пакет-транзакцію, знімає верхній шар шифрування “цибулини” пакета, формує новий контракт HTLC і передає пакет наступній вершині, отриманій з тимчасової маршрутизаційної таблиці.

Для того, щоб програмне забезпечення могло автоматично перемикатись між застосуванням оригінального та модифікованого, а також будь-яких інших методів маршрутизації, необхідно реалізувати агрегатний маршрутизатор, який складається з таблиці об’єктів, що задовольняють інтерфейс `Router`, ключами в якій є чисельні значення версій пакетів, і в свою чергу задовольняє цей інтерфейс наступним чином:

1. `LookupPath` – здійснює пошук в таблиці маршрутизаторів за версією пакета та викликає метод `LookupPath` відповідного маршрутизатора, інакше надсилає сигнал про помилку.

2. `SendTransaction` – здійснює пошук в таблиці маршрутизаторів за версією пакета та викликає метод `SendTransaction` відповідного маршрутизатора, інакше надсилає сигнал про помилку.
3. `HandlePacket` – здійснює пошук в таблиці маршрутизаторів за версією пакета та викликає метод `HandlePacket` відповідного маршрутизатора, інакше надсилає сигнал про помилку.

Архітектура модуля маршрутизації подана на рис. 4.1.

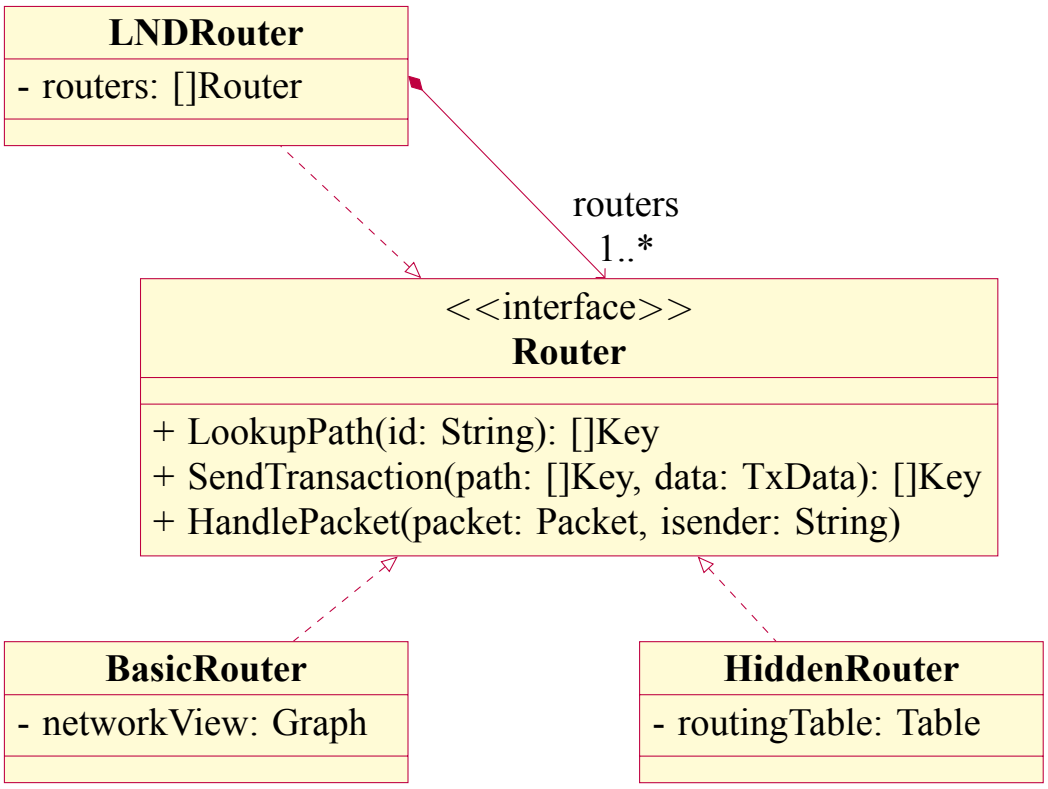


Рис. 4.1. Модуль маршрутизації

Основною сутністю, що відповідає за маршрутизацію, є об'єкт класу `LNDRouter`, який містить асоціативний масив маршрутизаторів і делегує виконання відповідних методів інтерфейсу `Router` конкретним маршрутизаторам на основі версії пакета, яка слугує ключем в асоціативному масиві.

Класи, що використовуються в розробленому програмному модулі для

відображення пакетів типу *Запит* та *Відповідь* подано на рис. 4.2.

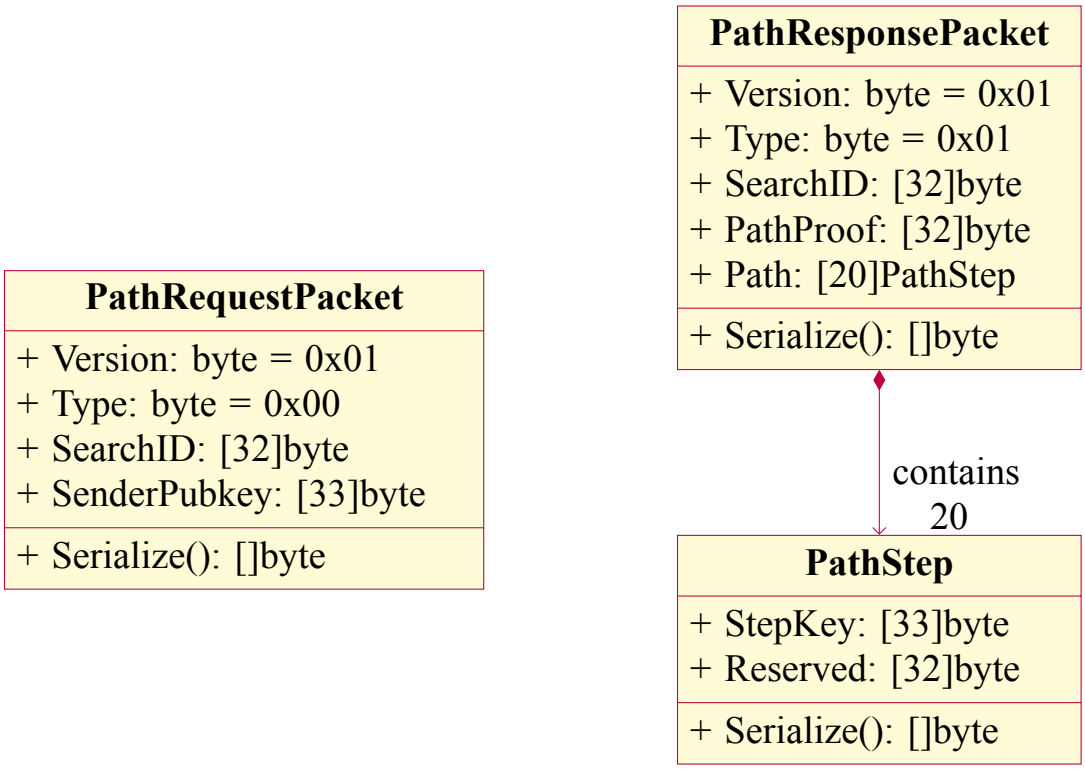


Рис. 4.2. Пакети пошуку/побудови шляху

4.3. Подальші дослідження

В процесі реалізації запропонованого методу було ідентифіковано декілька проблем та потенційних модифікацій, що дозволять усунути недоліки, пов'язані з додатковим навантаженням на мережу.

4.3.1. Попередження явища обриву шляху

Як вже було зазначено раніше, запропонована реалізація маршрутизації не передбачає механізму обмеження радіусу поширення пошукового пакета. Використовуючи дані про існуючу мережу транзакційних каналів Lightning, можна припустити, що максимального радіусу $L = 20$, встановленого протоколом, достатньо для покриття мережі, в декілька сотень разів більшої за розміром за поточну, але не зважаючи на це, відсутність механізму обмеження радіусу поширення

пошукового пакета призводить до того, що він обходить всю мережу, потрапляючи до кожної вершини, що входить в ту ж компоненту сильної зв'язності, до якої належить вихідна вершина. Два негативних наслідки такої особливості реалізації, як згадувалось раніше, полягають у тому, що:

1. Пошук шляху створює велику кількість надлишкового трафіку в мережі.
2. В умовах дуже великої мережі пошук шляху може повернути результат, що виходить за межі встановленого радіусу – так зване явище обриву шляху, – і в такому випадку процес надсилання транзакції завершиться помилкою, що не є проблемою з точки зору стійкості протоколу, але відтерміновує момент встановлення помилкового результату, що ще що може значно знизити його ефективність.

Тривіальні або традиційні підходи до вирішення цієї проблеми є субоптимальними, оскільки створюють додаткові метадані, що можуть використовуватись для аналізу відстаней між вершинами в мережі, що порушує властивість приховування топології. Таким чином, важливим напрямком подальших досліджень є розробка методу обмеження радіусу поширення пошукових пакетів, що не використовує лічильники TTL та інші вказівки на відстані між вершинами.

4.3.2. Усунення цибулевої маршрутизації

Основною причиною використання цибулевої маршрутизації в оригінальному протоколі є необхідність приховування від кожної вершини I_i на транзакційному шляху інформації про всі інші вершини, окрім попередньої та наступної вершин I_{i-1} та I_{i+1}. Втім, за умови використання тимчасових маршрутних вказівок, що зберігаються кожною вершиною на шляху, і мають форму (h, I_{i-1}, I_{i+1}), як це реалізовано у запропонованому методі маршрутизації, та за умови, що транзакційні

дані (кількість коштів) однакові для всіх вершин на шляху, можна зробити спостереження, що вся інформація, що потрібна для надсилання транзакції вже розподілена між вершинами на транзакційному шляху, а отже, цибулеву маршрутизацію транзакційних даних можна відкинути.

Основною модифікацією запропонованого методу маршрутизації в такому випадку є зміна структури пакету-відповіді: замість послідовності ключів вершина на транзакційному шляху він міститиме лише інформацію, необхідну для підтвердження існування шляху до кінцевої вершини:

$$Resp = (h, rP_S).$$

Транзакційний пакет в такому випадку повинен містити ідентифікатор h для здійснення пошуку в локальних маршрутизаційних таблицях вершин на шляху.

Важливим напрямком подальших дослідження є аналіз безпеки такого підходу, зокрема вивчення ризиків та можливостей захисту від “нечесної” маршрутизації пакета вершинами на транзакційному шляху. Окрім того, такий підхід повинен запропонувати емпіричні оцінки закінчення терміну дії тимчасових маршрутизаційних вказів, що не порушуватимусть стабільності роботи мережі.

4.3.3. Розміщення маршрутних вказівок в пакеті зі шляхом

Цибулеву маршрутизацію транзакційних пакетів оригінального протоколу можна розглядати як інструмент забезпечення самодостатньості пакетів – кожен транзакційний пакет містить всю необхідну інформацію для здійснення маршрутизації, і не потребує додаткової підтримки вершин на транзакційному шляху, за винятком створення нових контрактів HTLC. Якщо властивість самодостатньості необхідно зберегти, запропонований метод маршрутизації можна

модифікувати шляхом розміщення маршрутних вказівок в зашифрованому вигляді безпосередньо в пакеті-відповіді. Це дозволить уникнути підтримування локальних маршрутизаційних таблиць, що знизить залежність запропоноваго методу від цілісності даних у вершинах мережі, але при цьому значно збільшить розмір пакета.

Оскільки маршрутна вказівка формується з точки зору поточної вершини, вона містить інформацію про основний ключ, чи адресу, наступної вершини, а отже основний питанням цього напрямку дослідження уникнення витоку інформації про фактичні ключі учасників шляху. Одним з варіантів реалізації такого захисту є використання цибулевої маршрутизації та приховуючих множників, аналогічно до методу, описаного в пункті 2.2.2.

4.4. Висновки

В даному розділі розглядається основна властивість запропонованого методу маршрутизації, а саме властивість приховування топології, досягнення якої було головною ціллю даного дослідження, а також його недоліки порівняно з оригінальним методом (див. підрозділ 4.1).

Для перевірки властивості приховування топології мережі надано формальну модель зловмисника у вигляді функції існування шляху, що відображає спробу відновити топологію мережі з інформації про існування шляху між вершинами певної множини вершин та аргументовано, що такий зловмисник не може відновити структуру мережі з множини значень функції пошуку шляху, за умови, що він знає про існування лише невеликої підмножини вершин мережі.

Описано два основних недоліки запропонованого методу, а саме сповільнення процесу маршрутизації в середньому в три рази, що викликано двома додатковими проходженнями транзакційним маршрутом

в процесі надсилання транзакції, та збільшення обсягу даних, що передаються в мережі, через введення двох додаткових типів пакетів, для одного з яких не передбачено механізму обмеження радіусу поширення.

Описано структуру та основні компоненти виконаної в рамках дослідження програмної реалізації запропонованого методу на основі приватних транзакційних каналів у мережі Lightning та архітектура програмного забезпечення для виконання отриманого транзакційного протоколу в існуючій мережі. Окрім того, сформульовано напрямки подальших досліджень і вдосконалень запропонованого методу.

ВИСНОВКИ

В рамках даного дослідження було побудовано графову модель мережі криптовалютних каналів, на основі якої було сформульовано проблему витоку даних про фінансові зв’язки як наслідок відкритої топології такої мережі. Було продемонстровано, що наявність будь-якої інформації про зв’язок між вершиною мережі транзакційних каналів та сутностями реального світу дозволяє з певною ймовірністю формувати твердження про грошові потоки сусідніх вершин, що унеможливлює використання такої мережі в багатьох сферах повсякденного життя, оскільки порушує приватність фінансових даних її учасників.

В даному дослідженні запропоновано вирішення даної проблеми засобом формування мережі з неопублікованих, або приватних, транзакційних каналів. Така мережа забезпечує захист даних про фінансові зв’язки її учасників, оскільки обмежує доступ до інформації про ці зв’язки до пари вершин, між якими вони встановлені.

Для того, щоб у мережі приватних транзакційних каналів міг виконуватись транзакційний протокол – протокол передачі коштів між двома каналами, що належать одній і тій же вершині, було розроблено метод маршрутизації на основі тимчасових маршрутизаційних таблиць. Ключовими компонентами цього методу є алгоритми пошуку та конструювання шляху в мережі, який полягає в тому, що початкова вершина транзакційного шляху здійснює пошук в ширину в графі мережі, формуючи запит на пошук шляху до кінцевої вершини, надсилаючи його всім своїм безпосереднім сусідам і очікуючи відповіді. Отримана відповідь міститиме всю необхідну інформацію для здійснення машрутизації транзакції встановленим шляхом.

Для того, щоб приховати відстань від поточної вершини на шляху до початкової та кінцевої вершин, використовується фіксований розмір

послідовності ключів у відповіді на запит пошуку шляху. Для цього алгоритм конструювання шляху змушує кінцеву вершину створити максимально дозволену протоколом кількість одноразових ключів, а кожну вершину на шляху – відкидати останній ключ у послідовності і додавати свій власний ключ на її початок, здійснюючи таким чином зсув послідовності вправо на один елемент. Результатом цього є те, що початкова вершина отримує послідовність ключів вершин-посередників у порядку, визначеному топологією мережі, але оскільки ця послідовність має фіксовану довжину, а ключі в ній – одноразові, неможливо визначити фактичну відстань до кінцевої вершини. Аналогічне твердження справджується для всіх вершин на транзакційному шляху.

Для аналізу безпеки запропонованого методу було побудовано модель зловмисника на основі функції існування шляху в графі мережі. Виходячи з припущення, що зловмисникуу відома лише мала підмножина вершин в графі та множина значень функції існування шляху між будь-якими двома з цих вершин, можна стверджувати що структуру графа відновити неможливо, оскільки для будь-якого значення функції існування шляху, неможливо визначити, чи шлях є безпосереднім (що вказало б на існування каналу між двома відповідними вершинами). Таким чином, мережа приватних транзакційних каналів зберігає властивість приховування топології за умови використання запропонованого методу маршрутизації.

Також було запропоновано напрямки подальших дослідження, орієнтовані на покращення характеристик ефективності, розміру пакета та навантаження на мережу.

З метою тестування було здійснено реалізацію запропонованого методу маршрутизації засобом модифікації програмного забезпечення для роботи з існуючою мережею транзакційних каналів Lightning. Для цього було розроблено інтерфейс агрегатного маршрутизатора, який складається з об’єктів-маршрутизаторів, що реалізують доступні

методи маршрутизації, і делегує виконання необхідних методів до конкретних маршрутизаторів, обираючи їх на основі версії вхідного пакета. Розроблена реалізація дозволяє оператору вершини мережі активувати запропонований метод на вибір, і за умови наявності достатньої кількості приватних каналів у мережі, здійснювати маршрутизацію транзакцій, не виходячи за межі підмережі з прихованою топологією.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Nakamoto S. Bitcoin: A peer-to-peer electronic cash system. — 2008. — Access mode: http://bitcoin.org/bitcoin.pdf.
2. Poon J., Dryja T. The bitcoin lightning network: Scalable off-chain instant payments. — 2016. — Access mode: https://lightning.network/lightning-network-paper.pdf.
3. Dwork Cynthia; Naor M. Pricing via processing, or, combatting junk mail, advances in cryptology. — CRYPTO'92: Lecture Notes in Computer Science No. 740. Springer: 139–147. — 1993. — Access mode: http://www.wisdom.weizmann.ac.il/~naor/PAPERS/pvp.ps.
4. Back A. A partial hash collision based postage scheme. — Hashcash.org. Retrieved 13 October 2014. — 1997. — Access mode: http://www.hashcash.org/papers/announce.txt.
5. Lombrozo E., Lau J., Wuille P. Bip 141: Segregated witness (consensus layer). — 2015. — Access mode: https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki.
6. Lau J. Bip 114: Merkelized abstract syntax tree. — 2016. — Access mode: https://github.com/bitcoin/bips/blob/master/bip-0114.mediawiki.
7. Ben-Sasson E., Chiesa A., Garman C. et al. Zerocash: Decentralized anonymous payments from bitcoin. — Cryptology ePrint Archive, Report 2014/349. — 2014. — Access mode: https://eprint.iacr.org/2014/349.
8. Internet protocol : STD : 5 / RFC Editor ; Executor: Jon Postel : 1981. — Access mode: http://www.rfc-editor.org/rfc/rfc791.txt.
9. Danezis G., Goldberg I. Sphinx: A compact and provably secure mix format. — 2009.
10. Krawczyk D. H., Bellare M., Canetti R. HMAC: Keyed-Hashing for Message Authentication. — RFC 2104. — 1997. — Access mode: https://rfc-editor.org/rfc/rfc2104.txt.

11. Russel R. Bolt #4: Onion routing protocol. — 2017. — Access mode: https://github.com/lightningnetwork/lightning-rfc/blob/master/04-onion-routing.md.
12. Hypertext transfer protocol – http/1.1 : RFC : 2616 / RFC Editor ; Executor: Roy T. Fielding, James Gettys, Jeffrey C. Mogul et al. : 1999. — June. — Access mode: http://www.rfc-editor.org/rfc/rfc2616.txt. — http://www.rfc-editor.org/rfc/rfc2616.txt.
13. Hess P. Sec 2: Recommended elliptic curve domain parameters. — 2000.
14. Dingledine R., Mathewson N., Syverson P. Tor: The second-generation onion router. — 2004. — Access mode: http://dl.acm.org/citation.cfm?id=1251375.1251396.
15. The Bittorrent P2P File-Sharing System: Measurements and Analysis / Ed. by Miguel Castro, Robbert van Renesse. — Berlin, Heidelberg : Springer Berlin Heidelberg, 2005.
16. Osuntokun O., Fromknecht C., Halseth J. T. Lnd. — 2019. — Access mode: https://github.com/lightningnetwork/lnd.

ДОДАТКИ

Додаток 1

Копія презентації

Метод маршрутизації транзакцій в мережі криптовалютних каналів з прихованою топологією

Жикін Ю. С., магістрант
Науковий керівник: Онай М. В., к.т.н., доцент

22 травня, 2019

Актуальність

- Мережі транзакційних каналів є найбільш ефективним з точки зору пропускної здатності застосування криптовалютної технології Bitcoin.
- Основним процесом в такій мережі є маршрутизація транзакцій, оскільки саме вона визначає процес здійснення транзакцій.
- Існуюча реалізація такої мережі використовує фіксовану маршрутизацію, що зумовлює необхідність публічного доступу до топології мережі, яка в свою чергу містить інформацію про фінансові зв'язки між її учасниками.
- Для вирішення проблеми публічного доступу до топології мережі необхідно розробити метод маршрутизації, що не використовує топологію мережі.

Структура дослідження

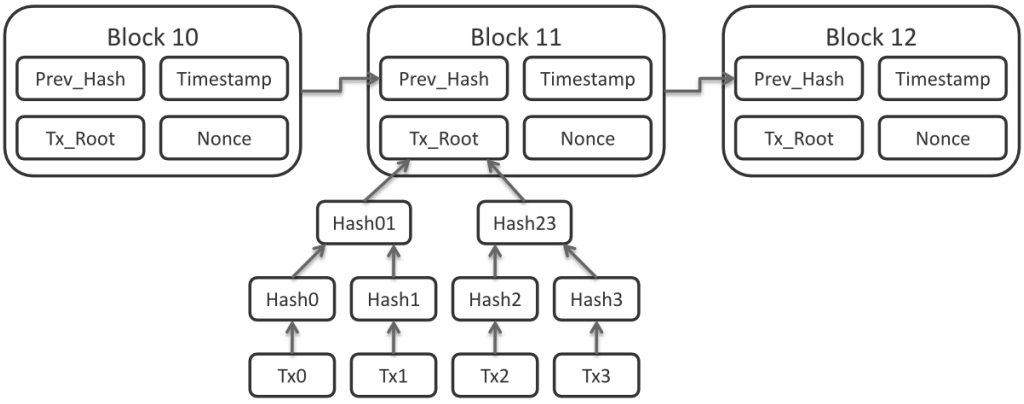
- **Об'єкт** – процес маршрутизації транзакції в мережі криптовалютних каналів в умовах відсутності даних про її топологію.
- **Предмет** – методи, алгоритми та програмна реалізація маршрутизації транзакцій в мережах криптовалютних каналів.
- **Методи** – моделювання мережі транзакційних каналів за допомогою теорії графів, композиція та адаптація загальних методів маршрутизації до предметної області, розробка програмного інтерфейсу маршрутизатора.

Мета та задачі дослідження

- **Мета** – побудувати модель мережі транзакційних каналів з прихованою топологією та розробити метод маршрутизації транзакцій у ній.
- **Задачі:**
  - побудувати модель мережі транзакційних каналів на основі теорії графів для формального опису проблематики,
  - сформулювати проблему відкритих фінансових зв'язків в мережі транзакційних каналів на основі графової моделі мережі,
  - розробити метод маршрутизації транзакцій в умовах відсутності інформації про топологію мережі,
  - розробити програмне забезпечення, що дозволяє використовувати різні методи маршрутизації в межах існуючої мережі транзакційних каналів Lightning.

Сучасна криптовалютна система

- Стаття та прототипна реалізація, Сатоші Накамото, 2008 р.
- Розподілена база даних історії транзакцій:
  - послідовність блоків з історією транзакцій, зв'язана за допомогою криптографічних хеш-функцій;
  - система "доказу роботи" *proof-of-work* – повного перебору рішень надскладної математичної задачі – захист від формування альтернативної історії.



Пропускна здатність

- Пропускна здатність T обмежена параметрами мережі:

$$T = \frac{S_B}{S_{tx} P_B} \text{ т./с},$$

де

S_B – розмір блока, у байтах,
S_{tx} – розмір транзакції, у байтах,
P_B – період між блоками, у секундах.

- Середня пропускна здатність мережі Біткоїн

$$T \approx 10^6 \text{ б}/(600 \text{ б} * 600 \text{ с}) \approx 2.7 \text{ т./с}$$

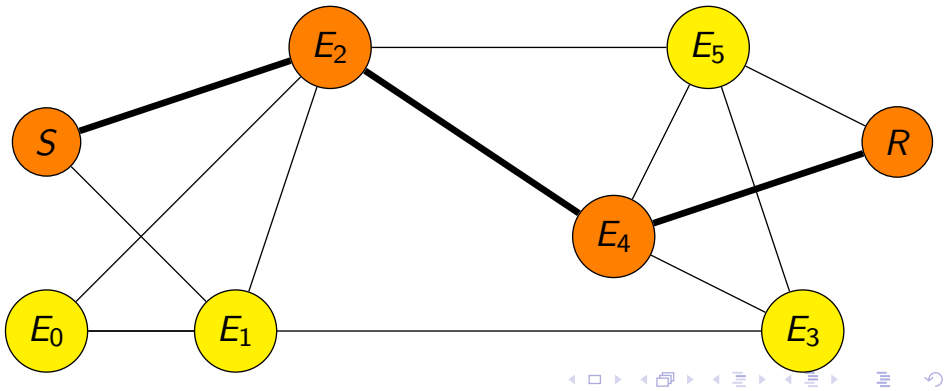
надзвичано мала, порівняно з заявленою середньою пропускною здатністю системи Visa (1700 т./с).

Криптовалютні транзакційні канали

- Використання основної криптовалютної мережі як інструменту фіналізації транзакцій.
- Протокол позамережевих транзакцій (транзакційний канал):
  - SETUP: $T_S \to S_0 = (a_0, b_0)$;
  - UPDATE: $S_i = (a_i, b_i) \to S_{i+1} = (a_i + d, b_i - d)$;
  - CLOSE: $S_i = (a_i, b_i) \to (T_{a_i}, T_{b_i})$.
- **Стислість** – результати алгоритму UPDATE не потрібно публікувати в основній мережі, що й дозволяє підвищити пропускну здатність.
- **Універсальна арбітрація** – правил консенсусу основної мережі достатньо для правильного завершення алгоритму CLOSE.

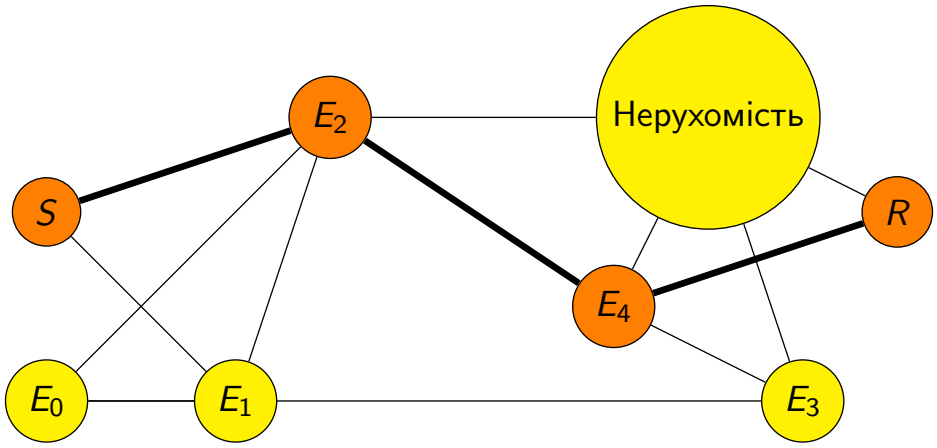
Мережа транзакційних каналів

- Множина сутностей, пов'язаних транзакційними каналами;
- Актори – вихідна вершина S, кінцева вершина R, вершини-посередники I_i, зовнішні вершини E_i.
- Транзакційний протокол – алгоритм перенесення коштів між двома каналами $((E_i, E_j), (E_j, E_k))$.
- Цибулева маршрутизація від джерела – вершина S будує шлях, створює "цибулину" з інформацією про наступні переходи і надсилає посереднику I_1.



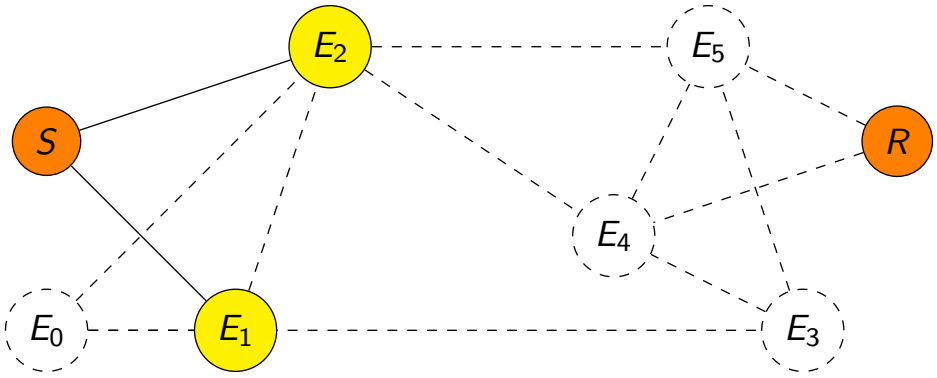
Мережа транзакційних каналів: відкрита топологія

- Дані про граф мережі є публічно доступними, оскільки використовуються для побудови шляху при маршрутизації.
- Наявність каналу вказує на існуючу фінансову взаємодію, що за наявності сторонніх даних може порушувати фінансову приватність учасників мережі.



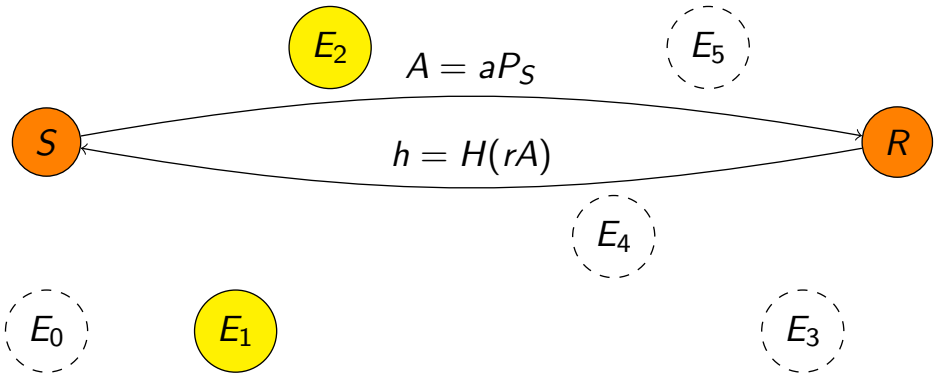
Мережа транзакційних каналів: прихована топологія

- Заміна маршрутизації "від джерела" використанням тимчасових маршрутизаційних таблиць дозволить усунути необхідність доступу до даних про топологію.
- Мережа з прихованою топологією забезпечує приватність інформації про фінансові зв'язки учасників мережі.



Запронований метод

- Етап 1 – ініціалізація:
  - вершина S генерує випадкове значення a і передає вершині R значення $A = rP_s$;
  - вершина R генерує значення r і передає вершині S значення хеш-функції $h = H(rA)$.

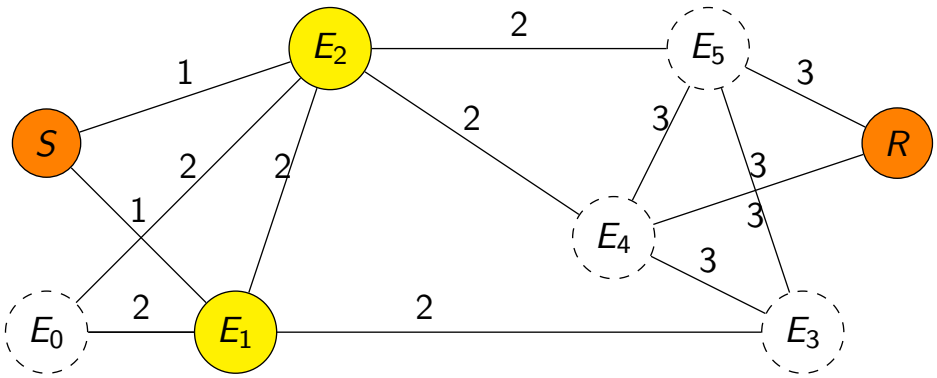


Запронований метод

- Етап 2 – пошук шляху:
  - вершина S передає своїм сусідам пошуковий пакет

$$Req = (h, P_S);$$

  - кожна вершина E_i в мережі, отримавши пакет, передає його своїм безпосереднім сусідам, якщо він не призначений для неї.



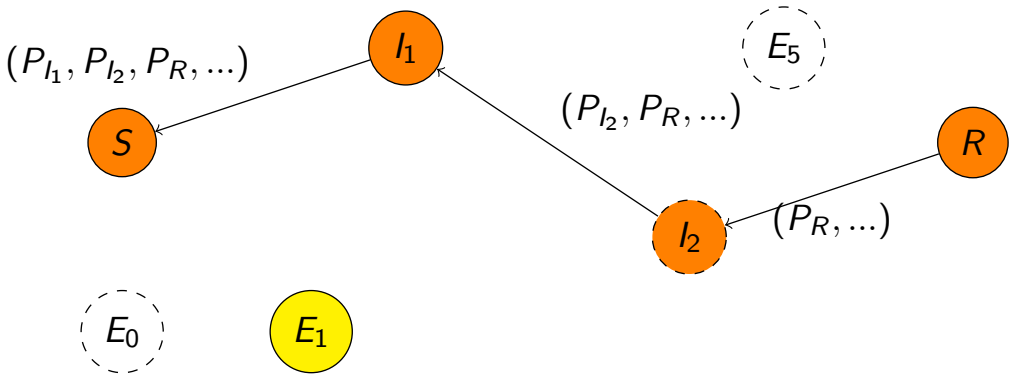
Запронований метод

- Етап 3 – побудова шляху:
  - вершина R, отримавши "свій" пошуковий пакет, генерує відповідь, що містить L точок еліптичної кривої:

$$Resp = (h, rP_S, P_R, P_1, ..., P_{L-1});$$

  - кожна вершина-посередник I_i додає до пакета свій ключ, зсуваючи послідовність ключів вправо і відкидаючи P_L:

$$Resp' = (h, R, P_I, P_1, ..., P_{L-1}).$$



Запронований метод

- Етап 4 – пересилання транзакції:
  - вершина S здійснює перевірку коректності:

$$H(R) = (aR) = H(arP_s) = H(rA) = h;$$

  - всі вершини-посередники I_i зберігають тимчасову маршрутизаційну таблицю:

$$M_{I_i}(h) = (I_{i-1}, I_{i+1});$$

  - початкова вершина використовує отриману послідовність ключів для формування пакета та здійснення пересилання.

Аналіз приватності фінансових зв'язків

- Приховування топології:
  - функція існування шляху $Q_{\mathcal{G}} : \mathcal{N} \times \mathcal{N} \mapsto \{0, 1\}$:

$$Q_{\mathcal{G}}(n_i, n_j) = \begin{cases} 1, & \text{якщо між } n_i \text{ та } n_j \text{ існує шлях у } \mathcal{G}, \\ 0 & \text{в усіх інших випадках;} \end{cases}$$

  - аналіз топології мережі зводиться до задачі відновлення структури графа $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, маючи множину вершин $\mathcal{N}' \subset \mathcal{N}$ і підмножину образу функції $\{Q_{\mathcal{G}}(n_i, n_j); n_i, n_j \in \mathcal{G}'\}$;
  - решта вершин графа $\mathcal{N} \setminus \mathcal{N}'$ є невідомою, тому для будь-якого відомого значення функції $Q_{\mathcal{G}}(n_i, n_j) = 1$ неможливо визначити безпосередність зв'язку (існування ребра (n_i, n_j)).

Формулювання недоліків та подальші дослідження

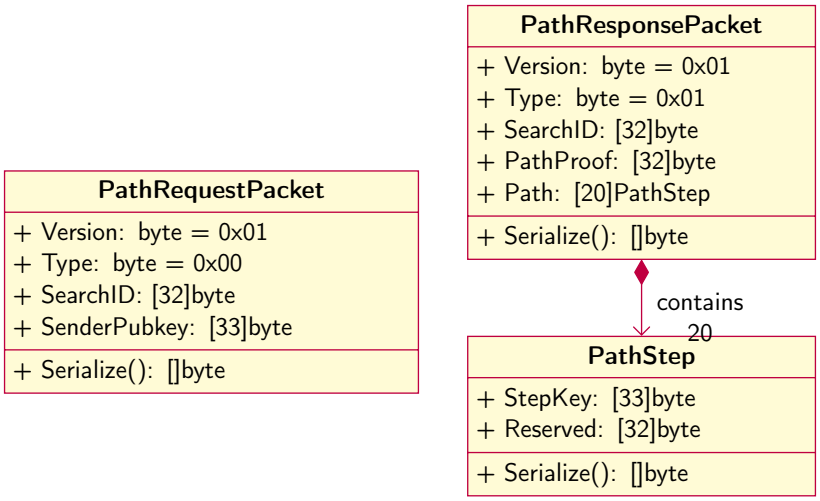
- Недоліки:
  - запропонований метод в середньому є у 3 рази повільнішим, ніж метод маршрутизації “від джерела”, оскільки побудова шляху вимагає 2-х додаткових проходжень цим шляхом;
  - паралельний пошук шляху в ширину призводить до значного підвищення трафіку в мережі.
- Подальші дослідження:
  - розробка методу обмеження пропагації пошукового пакета, що не порушує приховування топології;
  - дослідження можливості включення одноелементних тимчасових маршрутизаційних таблиць безпосередньо у пакеті-відповіді, що зробить транзакційним пакета самодостатнім.

Програмна реалізація запропонованого методу

- LND – основна реалізація набору протоколів Lightning з використанням програмного середовища Go.
- Мережа транзакційних каналів з прихованою топологією побудована з приватних транзакційних каналів, що використовуються в мережі Lightning для безпосередніх розрахунків.
- Описаний вище протокол маршрутизації доданий як опція конфігурації для існуючого програмного забезпечення мережі Lightning.

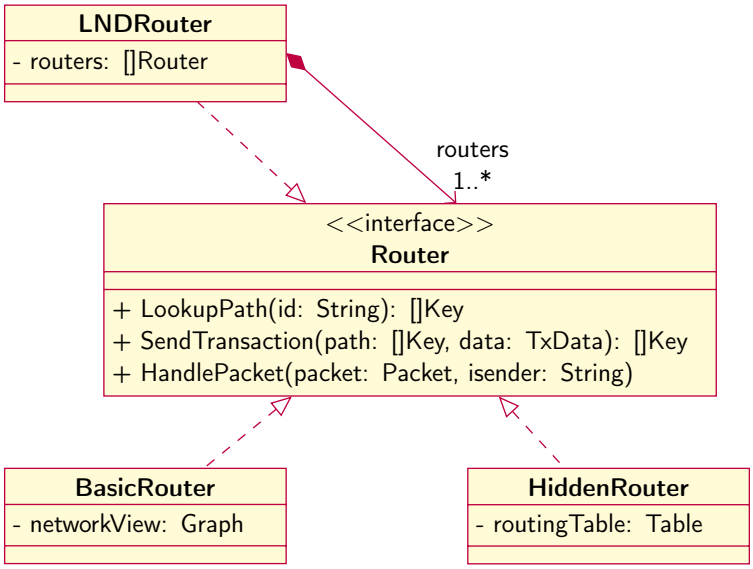
Програмна реалізація запропонованого методу

- Пакети даних, що використовуються на етапах пошуку та побудови шляху.



Програмна реалізація запропонованого методу

- Агрегація доступних методів маршрутизації, що вибирає потрібний маршрутизатор на основі типу пакета.



Апробація

- *Адаптований спрощений протокол транзакційних каналів для криптовалютної системи Monero* – Онай М.В., Жикін Ю.С., XI наукова конференція магістрантів та аспірантів "Прикладна математика та комп'ютинг" (ПМК-2018-2), Збірник тез доповідей, ст. 233-237, 2018.
- *Blind Payment Protocol for Payment Channel Networks* – Zhengbing Hu, Ivan Dychka, Mykola Onai, Yuri Zhykin, International Journal of Computer Network and Information Security (IJCNIS), MECS Press – Hong-Kong, accepted, May, 2019.

Наукова новизна

- Розроблено модель мережі криптовалютних каналів з прихованою топологією, яка відрізняється від існуючої моделі тим, що не містить приховує інформацію про кількість вершин в мережі та зв'язки між ними.
- Запропоновано метод маршрутизації транзакцій в мережі криптовалютних каналів, який відрізняється від існуючих тим, що не потребує доступу до інформації про топологію мережі, таким чином дозволяючи приховати цю інформацію і покращити характеристики приватності фінансових даних.

Висновки

- Побудовано графову модель акторів мережі криптовалютних каналів, що дозволяє
  - описати та візуалізувати протоколи взаємодії між учасниками мережі з точки зору функцій, які вони виконують;
  - сформулювати поняття транзакційної топології – структури мережі, що розглядається в контексті семантики зв'язків між вершинами;
  - формалізувати проблему відкритої транзакційної топології.
- Розроблено модель мережі криптовалютних каналів з прихованою топологією, яка
  - за визначенням приховує транзакційну топологію, а отже покращує характеристики приватності фінансових зв'язків учасників;
  - може складатись з неопублікованих криптовалютних каналів та бути прихованою підмережею існуючої публічної мережі.

Висновки

- Розроблено метод маршрутизації транзакцій, що
  - замінює фіксовану маршрутизацію від джерела динамічною маршрутизацією на основі тимчасових маршрутизаційних таблиць;
  - не потребує доступу до даних про топологію всієї мережі, і тому дозволяє повністю її приховати;
  - приховує довжину шляху на етапі пошуку шляху та збору необхідних ключів.
- Розроблено тестову реалізацію запропонованого методу, яка
  - працює в межах підмережі приватних транзакційних каналів в існуючій мережі Lightning;
  - динамічно обирає метод маршрутизації залежно від типу пакета даних, що обробляється;
  - виділена в окремий програмний інтерфейс, що передбачає можливість доповнення існуючого програмного забезпечення іншими методами маршрутизаці.

Дякую за увагу!

Додаток 2

Текст програми

```
package routing

import proto "github.com/golang/protobuf/proto"
import fmt "fmt"
import math "math"

// Reference imports to suppress errors if they are not otherwise used.
var _ = proto.Marshal
var _ = fmt.Errorf
var _ = math.Inf

// This is a compile-time assertion to ensure that this file
// is compatible with the proto package it is being compiled against.
// A compilation error at this line likely means your copy of the
// proto package needs to be updated.
const _ = proto.ProtoPackageIsVersion2

type PathRequestPacket struct {
  Version      []byte `protobuf:"bytes,1,opt,name=Version,proto3"
  json:"Version,omitempty"`
  Type         []byte `protobuf:"bytes,2,opt,name=Type,proto3"
  json:"Type,omitempty"`
  SearchID     []byte `protobuf:"bytes,3,opt,name=SearchID,proto3"
  json:"SearchID,omitempty"`
  SenderPubKey []byte `protobuf:"bytes,4,opt,name=SenderPubKey,proto3"
  json:"SenderPubKey,omitempty"`
}

func (m *PathRequestPacket) Reset() {
  *m = PathRequestPacket{}
}

func (m *PathRequestPacket) String() string {
  return proto.CompactTextString(m)
}

func (*PathRequestPacket) ProtoMessage() {}

func (*PathRequestPacket) Descriptor() ([]byte, []int) {
  return fileDescriptor0, []int{0}
```

```
}

func (m *PathRequestPacket) GetVersion() []byte {
  if m != nil {
    return m.Version
  }
  return nil
}

func (m *PathRequestPacket) GetType() []byte {
  if m != nil {
    return m.Type
  }
  return nil
}

func (m *PathRequestPacket) GetSearchID() []byte {
  if m != nil {
    return m.SearchID
  }
  return nil
}

func (m *PathRequestPacket) GetSenderPubKey() []byte {
  if m != nil {
    return m.SenderPubKey
  }
  return nil
}

type PathResponsePacket struct {
  Version   []byte    `protobuf:"bytes,1,opt,name=Version,proto3"
  json:"Version,omitempty"`
  Type      []byte    `protobuf:"bytes,2,opt,name=Type,proto3"
  json:"Type,omitempty"`
  SearchID  []byte    `protobuf:"bytes,3,opt,name=SearchID,proto3"
  json:"SearchID,omitempty"`
  PathProof []byte    `protobuf:"bytes,4,opt,name=PathProof,proto3"
  json:"PathProof,omitempty"`
  Path      *PathStep `protobuf:"bytes,5,opt,name=Path"
```

```
  json:"Path,omitempty"`
}

func (m *PathResponsePacket) Reset() {
  *m = PathResponsePacket{}
}

func (m *PathResponsePacket) String() string {
  return proto.CompactTextString(m)
}

func (*PathResponsePacket) ProtoMessage() {}

func (*PathResponsePacket) Descriptor() ([]byte, []int) {
  return fileDescriptor0, []int{1}
}

func (m *PathResponsePacket) GetVersion() []byte {
  if m != nil {
    return m.Version
  }
  return nil
}

func (m *PathResponsePacket) GetType() []byte {
  if m != nil {
    return m.Type
  }
  return nil
}

func (m *PathResponsePacket) GetSearchID() []byte {
  if m != nil {
    return m.SearchID
  }
  return nil
}

func (m *PathResponsePacket) GetPathProof() []byte {
  if m != nil {
```

```
    return m.PathProof
  }
  return nil
}

func (m *PathResponsePacket) GetPath() *PathStep {
  if m != nil {
    return m.Path
  }
  return nil
}

type PathStep struct {
  StepKey  []byte `protobuf:"bytes,1,opt,name=StepKey,proto3"
  json:"StepKey,omitempty"`
  Reserved []byte `protobuf:"bytes,2,opt,name=Reserved,proto3"
  json:"Reserved,omitempty"`
}

func (m *PathStep) Reset() {
  *m = PathStep{}
}

func (m *PathStep) String() string {
  return proto.CompactTextString(m)
}

func (*PathStep) ProtoMessage() {}

func (*PathStep) Descriptor() ([]byte, []int) {
  return fileDescriptor0, []int{2}
}

func (m *PathStep) GetStepKey() []byte {
  if m != nil {
    return m.StepKey
  }
  return nil
}
```

```
func (m *PathStep) GetReserved() []byte {
  if m != nil {
    return m.Reserved
  }
  return nil
}

func init() {
  proto.RegisterType((*PathRequestPacket)(nil),
  "routing.PathRequestPacket")
  proto.RegisterType((*PathResponsePacket)(nil),
  "routing.PathResponsePacket")
  proto.RegisterType((*PathStep)(nil),
  "routing.PathStep")
}

func init() {
  proto.RegisterFile("packet.proto", fileDescriptor0)
}

// generateSphinxPacket generates then encodes a sphinx packet which encodes
// the onion route specified by the passed layer 3 route. The blob returned
// from this function can immediately be included within an HTLC add packet
// to be sent to the first hop within the route.
func generateSphinxPacket(route *Route, paymentHash []byte) ([]byte,
*sphinx.Circuit, error) {
  // First obtain all the public keys along the route which are contained
  // in each hop.
  nodes := make([]*btcec.PublicKey, len(route.Hops))
  for i, hop := range route.Hops {
    // We create a new instance of the public key to avoid possibly
    // mutating the curve parameters, which are unset in a higher
    // level in order to avoid spamming the logs.
    pub := btcec.PublicKey{
      Curve: btcec.S256(),
      X:     hop.Channel.Node.PubKey.X,
      Y:     hop.Channel.Node.PubKey.Y,
    }
    nodes[i] = &pub
  }
```

```
// Next we generate the per—hop payload which gives each node within
// the route the necessary information (fees, CLTV value, etc) to
// properly forward the payment.
hopPayloads := route.ToHopPayloads()

log.Tracef("Constructed per—hop payloads for payment_hash=%x: %v",
paymentHash[:], spew.Sdump(hopPayloads))

sessionKey, err := btcec.NewPrivateKey(btcec.S256())
if err != nil {
  return nil, nil, err
}

// Next generate the onion routing packet which allows us to perform
// privacy preserving source routing across the network.
sphinxPacket, err := sphinx.NewOnionPacket(nodes, sessionKey,
hopPayloads, paymentHash)
if err != nil {
  return nil, nil, err
}

// Finally, encode Sphinx packet using it's wire representation to be
// included within the HTLC add packet.
var onionBlob bytes.Buffer
if err := sphinxPacket.Encode(&onionBlob); err != nil {
  return nil, nil, err
}

log.Tracef("Generated sphinx packet: %v",
newLogClosure(func() string {
  // We unset the internal curve here in order to keep
  // the logs from getting noisy.
  sphinxPacket.EphemeralKey.Curve = nil
  return spew.Sdump(sphinxPacket)
}),
)

return onionBlob.Bytes(), &sphinx.Circuit{
  SessionKey:  sessionKey,
```

```
    PaymentPath: nodes,
  }, nil
}

// LightningPayment describes a payment to be sent through the network
// to the final destination.
type LightningPayment struct {
  // Target is the node in which the payment should be routed towards.
  Target *btcec.PublicKey

  // Amount is the value of the payment to send through the network in
  // milli-satoshis.
  Amount lnwire.MilliSatoshi

  // PaymentHash is the r-hash value to use within the HTLC extended to
  // the first hop.
  PaymentHash [32]byte
}

// SendPayment attempts to send a payment as described within the passed
// LightningPayment. This function is blocking and will return either:
// when the payment is successful, or all candidates routes have been
// attempted and resulted in a failed payment. If the payment succeeds,
// then a non-nil Route will be returned which describes the path the
// successful payment traversed within the network to reach the
// destination. Additionally, the payment preimage will also be returned.
func (r *ChannelRouter) SendPayment(payment *LightningPayment)
([32]byte, *Route, error) {
  log.Tracef("Dispatching route for lightning payment: %v",
  newLogClosure(func() string {
    payment.Target.Curve = nil
    return spew.Sdump(payment)
  }),
  )

  var (
  preImage  [32]byte
  sendError error
  )
```

```
// First, we'll kick off the payment attempt by attempting to query the
// known channel graph for a route to the destination.
routes, err := r.FindRoutes(payment.Target, payment.Amount)
if err != nil {
  return preImage, nil, err
}

// For each eligible path, we'll attempt to successfully send our
// target payment using the multi—hop route. We'll try each route
// serially until either once succeeds, or we've exhausted our set of
// available paths.
for _, route := range routes {
  log.Tracef("Attempting to send payment %x, using route: %v",
  payment.PaymentHash, newLogClosure(func() string {
    return spew.Sdump(route)
  }),
  )

  // Generate the raw encoded sphinx packet to be included along
  // with the htlcAdd message that we send directly to the
  // switch.
  onionBlob, circuit, err := generateSphinxPacket(route,
  payment.PaymentHash[:])
  if err != nil {
    return preImage, nil, err
  }

  // Craft an HTLC packet to send to the layer 2 switch. The
  // metadata within this packet will be used to route the
  // payment through the network, starting with the first—hop.
  htlcAdd := &lnwire.UpdateAddHTLC{
    Amount:      route.TotalAmount,
    Expiry:      route.TotalTimeLock,
    PaymentHash: payment.PaymentHash,
  }
  copy(htlcAdd.OnionBlob[:], onionBlob)

  // Attempt to send this payment through the network to complete
  // the payment. If this attempt fails, then we'll continue on
  // to the next available route.
```

```
firstHop := route.Hops[0].Channel.Node.PubKey
preImage, sendError = r.cfg.SendToSwitch(firstHop, htlcAdd,
circuit)
if sendError != nil {
  // An error occurred when attempting to send the
  // payment, depending on the error type, we'll either
  // continue to send using alternative routes, or simply
  // terminate this attempt.
  log.Errorf("Attempt to send payment %x failed: %v",
  payment.PaymentHash, sendError)

  switch onionErr := sendError.(type) {
    // If the end destination didn't know they payment
    // hash, then we'll terminate immediately.
    case *lnwire.FailUnknownPaymentHash:
    return preImage, nil, sendError

    // If we sent the wrong amount to the destination, then
    // we'll exit early.
    case *lnwire.FailIncorrectPaymentAmount:
    return preImage, nil, sendError

    // If the time-lock that was extended to the final node
    // was incorrect, then we can't proceed.
    case *lnwire.FailFinalIncorrectCltvExpiry:
    return preImage, nil, sendError

    // If we crafted an invalid onion payload for the final
    // node, then we'll exit early.
    case *lnwire.FailFinalIncorrectHtlcAmount:
    return preImage, nil, sendError

    // Similarly, if the HTLC expiry that we extended to
    // the final hop expires too soon, then will fail the
    // payment.
    //
    // again with recent block height
    case *lnwire.FailFinalExpiryTooSoon:
    return preImage, nil, sendError
```

```
// If we erroneously attempted to cross a chain border,
// then we'll cancel the payment.
case *lnwire.FailInvalidRealm:
return preImage, nil, sendError

// If we get a notice that the expiry was too soon for
// an intermediate node, then we'll exit early as the
// expected block height as shifted from underneath us.
case *lnwire.FailExpiryTooSoon:
update := onionErr.Update
if err := r.applyChannelUpdate(&update); err != nil {
  return preImage, nil, err
}
return preImage, nil, sendError

// If we hit an instance of onion payload corruption or
// an invalid version, then we'll exit early as this
// shouldn't happen in the typical case.
case *lnwire.FailInvalidOnionVersion:
return preImage, nil, sendError
case *lnwire.FailInvalidOnionHmac:
return preImage, nil, sendError
case *lnwire.FailInvalidOnionKey:
return preImage, nil, sendError

// If the onion error includes a channel update, and
// isn't necessarily fatal, then we'll apply the update
// an continue with the rest of the routes.
//
case *lnwire.FailAmountBelowMinimum:
update := onionErr.Update
if err := r.applyChannelUpdate(&update); err != nil {
  return preImage, nil, err
}
continue
case *lnwire.FailFeeInsufficient:
update := onionErr.Update
if err := r.applyChannelUpdate(&update); err != nil {
  return preImage, nil, err
}
```

```
continue
case *lnwire.FailIncorrectCltvExpiry:
update := onionErr.Update
if err := r.applyChannelUpdate(&update); err != nil {
  return preImage, nil, err
}
continue
case *lnwire.FailChannelDisabled:
update := onionErr.Update
if err := r.applyChannelUpdate(&update); err != nil {
  return preImage, nil, err
}
continue
case *lnwire.FailTemporaryChannelFailure:
update := onionErr.Update
if err := r.applyChannelUpdate(update); err != nil {
  return preImage, nil, err
}
continue

// If the send fail due to a node not having the
// required features, then we'll note this error and
// continue
case *lnwire.FailRequiredNodeFeatureMissing:
continue

// If the send fail due to a node not having the
// required features, then we'll note this error and
// continue
//
case *lnwire.FailRequiredChannelFeatureMissing:
continue

// If the next hop in the route wasn't known or
// offline, we'll note this and continue with the rest
// of the routes.
//
case *lnwire.FailUnknownNextPeer:
continue
```

```
            // If the node wasn't able to forward for which ever
            // reason, then we'll note this and continue with the
            // routes.
            case *lnwire.FailTemporaryNodeFailure:
            continue

            // If we get a permanent channel or node failure, then
            // we'll note this (exclude the vertex/edge), and
            // continue with the rest of the routes.
            case *lnwire.FailPermanentChannelFailure:
            continue
            case *lnwire.FailPermanentNodeFailure:
            continue

            default:
            return preImage, nil, sendError
          }
        }

        return preImage, route, nil
      }

      // If we're unable to successfully make a payment using any of the
      // routes we've found, then return an error.
      return [32]byte{}, nil, fmt.Errorf("unable to route payment to "+
      "destination: %v", sendError)
    }
```